VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2021 Bc. David Kohout

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

IMPLEMENTACE ZABEZPEČENÍ DO DLMS PROTOKOLU
DLMS SECURITY SUITE IMPLEMENTATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. David Kohout**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Petr Mlýnek, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. David Kohout ***ID:*** 195823
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Implementace zabezpečení do DLMS protokolu

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte problematiku vzdáleného měření a využívaných protokolů a standardů pro toto měření. Proveďte analýzu kybernetické bezpečnosti v oblasti Smart Metering se zaměřením na protokol DLMS. Proveďte návrh implementace Security Suite v1 a v2 do knihovny Gurux. Analyzujte knihovnu Gurux DLMS/COSEM a proveďte rozšíření této knihovny a implementaci Security Suite v1 a v2 do zvoleného zařízení (např. Raspberry Pi).

DOPORUČENÁ LITERATURA:

[1] Gurux.DLMS | Gurux for DLMS smart meters. [online]. [cit. 2018-09-13]. Dostupné z: http://www.gurux.fi/Gurux.DLMS

[2] SOOD, J.K., D. FISCHER, J.M. EKLUND a T. BROWN. Developing a communication infrastructure for the smart grid. IEEE Electrical power & energy conference, 2009. Dostupné z: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5420809>.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: doc. Ing. Petr Mlýnek, Ph.D.
Konzultant: Ing. Jan Dvořák (ZPA)

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Práce se zaměřuje na problematiku zabezpečení pro chytré elektroměry podle specifikace DLMS Security Suite. Kromě využití standardu je v práci popsaná problematika adresování, jako jsou klientské a serverové adresy, včetně adresování pomocí sériového čísla. Dále jsou na konkrétní ukázce rozebrány způsoby formátování zpráv.
V části zabezpečení jsou představeny způsoby autentizace a šifrování zpráv. S bezpečností také souvisí zabezpečovací sady, které konkretizují použité algoritmy a velikosti klíčů.
Práce se také zabývá vývojem testovací aplikace s názvem VUT DLMS Tester, která se využívá k testování tohoto standardu pro reálné využití v distribuční soustavě. Celý vývoj je konzultován a kooperován s distribučními společnostmi a výrobci elektroměrů. Ze strany zabezpečení je aplikace rozšířená o možnosti využití vyšších úrovní zabezpečení, které DLMS podporuje. Pomocí aplikace jsou otestované různé scénáře vyčítání dat z elektroměrů s ohledem na datový nárůst při využití zabezpečení.

KLÍČOVÁ SLOVA

DLMS, chytrý elektroměr, Security Suite, testování, zabezpečení

ABSTRACT

This thesis is focused on smart meters cyber security using DLMS Security Suite standard. Security is more important every day and DLMS specifies multiple methods of authentication and message encryption which is defined in Security Suites. For message creation there are multiple possible formats that can be used. For a better understanding all formats are analysed and compared using a specific example.
Main part of this work is about making a testing application called VUT DLMS Tester which is used for testing this standard for real usage in distribution network. Whole development is cooperated with distribution companies and smart meter manufacturers. Application also implements higher levels of security which DLMS specifies. This application is used for testing multiple scenarios regarding security impact on data volume.

KEYWORDS

DLMS, security, Security Suite, smart meter, testing

KOHOUT, David. *Implementace zabezpečení do DLMS protokolu*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2021, 58 s. Diplomová práce. Vedoucí práce: doc. Ing. Petr Mlýnek, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. David Kohout
VUT ID autora:	195823
Typ práce:	Diplomová práce
Akademický rok:	2020/2021
Téma závěrečné práce:	Implementace zabezpečení do DLMS protokolu

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Petru Mlýnkovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Tato diplomová práce se zaměřuje na specifikaci DLMS/COSEM, kde popisuje některé základy a především se věnuje oblasti zabezpečení tohoto standardu.

V práci je nejprve popsáno využití DLMS. Následně jsou detailněji popsané některé základní prvky DLMS jako je především způsob adresování. Dále je rozebrána oblast zabezpečení samotného protokolu, včetně různých autentizačních metod a různých druhů zabezpečení, které jsou specifikované pomocí zabezpečovacích sad, včetně použitých algoritmů a certifikátů. Následně jsou na konkrétním příkladu znázorněné možné typy formátování zpráv, které se v DLMS používají, a také jsou tato data analyzována s porovnáním mezi používanými formáty.

Předposlední kapitola je již zaměřena na knihovnu Gurux a na vývoj aplikace pro testování DLMS, která nese název „VUT DLMS Tester“. Tato aplikace také implementuje zmiňované zabezpečení, včetně Security Suite 1 a 2. S touto aplikací byla vyvinuta také aplikace pro stranu serveru (elektroměru), která umožňuje emulovat jakýkoliv reálný elektroměr. Obě aplikace jak pro klientskou stranu, tak i stranu serveru je možné spustit na jakémkoliv operačním systému, na kterém je nainstalovaná Java 11. Pro testování v této práci byla zvolena platforma Raspberry Pi.

Poslední kapitola popisuje testování různých scénářů zabezpečení na konkrétním případu čtených objektů s využitím testovací aplikace. Toto testování je zaměřeno na porovnání datových objemů vzhledem k využitému zabezpečení.

1 Využití DLMS

V dnešní době, kdy se stále více objevují pojmy jako jsou IoT, chytré sítě a 5G je potřeba myslet i na další věci kolem nás. Kromě vysokorychlostního internetu, který využíváme v mobilních telefonech, je možné část kapacit věnovat i na měření energií. Přesně k tomuto účelu je určena specifikace DLMS.

Standard DLMS umožňuje využívat zařízení označované jako Smart Metery (chytré měřiče/metery). Do jednoho z nejvyužitelnějších odvětví pro chytré měření patří rozhodně elektroměry a tím tedy měření elektrické energie. Samotná specifikace umožňuje měření dalších asi 6 druhů energií, jako je například plyn, voda (teplá i studená), topení, chlazení či další média. Každá energie má specifikované vlastní objekty pro měření. Tato práce se zaměřuje především na měření elektrické energie.

DLMS nespočívá pouze v dálkových odečtech, kde by byla komunikace pouze jednosměrná, ale umožňuje vzájemnou komunikaci mezi meterem a řídícím centrem. Tím je tedy možné vzdáleně přepínat tarify, automaticky odpojovat odběrná místa, či provádět samotné vyúčtování (meter počítá spotřebu a zná tarif). Dále umožňuje spotřebiteli sledovat vlastní spotřebu v reálném čase.

Mezi hlavní důvody využívání těchto chytrých meterů určitě patří samotný monitoring sítě. Díky sledování všech výrobních i odběrných míst je možné interaktivněji reagovat na jakékoliv možné výkyvy v elektrické síti.

V České republice se již zhruba 40-50 let využívá technologie HDO (hromadné dálkové ovládání). Tento systém by se dal označit jako počátek chytrých sítí, avšak umožňoval pouze přepínání mezi nízkým a vysokým tarifem a dále umožňoval zasílat povel pro zapnutí/vypnutí některých elektrických spotřebičů (např. bojler a elektrické topení).

S nástupem chytrého měření (Smart Metering) má také napomoci nová vyhláška o měření elektřiny č. 359/2020 Sb [1], která nově od roku 2024 nařizuje distributorům instalovat chytré elektroměry na všechna odběrná místa s roční spotřebou vyšší než 6 MWh. Do této spotřeby patří zejména větší odběratelé, ale také domácnosti vytápěné elektřinou. Vyhláška také v příloze č. 4 určuje požadavky na zabezpečení přenášených dat. Z těchto požadavků stojí za zmínku využití `AES-256`, což znamená využít aktuálně nejvyšší úroveň zabezpečení, které DLMS umožňuje (kapitola 3.4.2).

V České Republice již existuje několik projektů, kde dochází k pilotnímu nasazování Smart Metering technologií. Mezi tyto projekty patří například Smart region Vrchlabí [2], případně projekt s názvem SMARAGD [3].

1.1 Výhody a nevýhody chytrých elektroměrů

Mezi výhody lze zařadit možné dálkové odečty, automatické vyúčtování, dálkové řízení, přepínání tarifů, monitoring a účinnější reakce na výkyvy.

Hlavní nevýhodou je vyšší pořizovací cena chytrých elektroměrů oproti běžným „hloupým“ elektroměrům. Tuto nevýhodu však lze kompenzovat ušetřením nákladů za manuální odečty a dále lze ušetřit na HDO, které již není potřebné jako samostatné zařízení, jelikož chytrý elektroměr dokáže nahradit tuto funkčnost.

Pokud porovnáme DLMS s technologií HDO, tak je možné pozorovat několik zásadních rozdílů. Hlavním rozdílem je množství funkcí, které DLMS dokáže poskytnout. HDO v podstatě dokáže provést pouze 2 funkce, a to zapnutí/vypnutí určitého spotřebiče či přepnutí tarifu. Tato funkčnost je jen zlomkem možností, které specifikace DLMS umožňuje.

Dalším rozdílem je standardizace. Technologie HDO není řízena žádnou normou či standardem. Naopak protokol DLMS je spravován asociací DLMS UA [4] a je specifikován v tzv. Barevných knížkách (Coloured Books [5, 6]) a v normě IEC 62056 [7]. Tím, že je technologie DLMS řízena standardem, je umožněna větší spolupráce mezi zařízeními různých výrobců a umožňuje také větší rozšíření samotné technologie.

Jednou z nevýhod DLMS je značný nárůst datového objemu oproti HDO. Avšak při porovnání celkového objemu dat k objemům v dnešních sítích je tato velikost zanedbatelná. Problém s datovým objemem však může nastat při použití přenosových technologií, které nemají tak velkou propustnost, jako například ZigBee, LoRaWAN, SigFox či NB-IoT.

Shrnutí výhod a nevýhod využití DLMS:

- Výhody
  - Dálkové odečty
  - Monitoring sítě
  - Automatické vyúčtování
  - Zastání funkčnosti HDO (přepínání tarifů)
  - Standardizace → rozšíření
  - Nezávislé na přenosové technologii
- Nevýhody
  - Vyšší cena oproti běžnému elektroměru
  - Velikost datového objemu pro některé technologie

2 Základy DLMS

Tato kapitola popisuje některé základní poznatky o DLMS, které nebyly obsaženy v bakalářské práci [8], či poskytnuté informace nebyly kompletní.

2.1 Adresování v DLMS

V jakýchkoliv síťových technologiích je velmi důležité správně určit, kam má být daná zpráva odeslána. K tomuto účelu slouží různé typy adres. V DLMS se kromě klasických adres (jako například IP nebo MAC adresa) závislých na přenosových technologiích, využívá další typ adresování. Tyto adresy se ve formátování zpráv (popsáno v kapitole 4.1) označují jako zdrojová a cílová adresa, avšak v rámci DLMS se označují následovně:

- Client Address
- Server Address

2.1.1 Client Address

`Client Address`, lze se také setkat s označením `ClientID` nebo `Client_SAP`, je velmi důležitým prvkem při konfiguraci spojení s meterem. Pomocí této adresy lze specifikovat k jaké sadě objektů se na meteru chceme připojit. Většinou je s klientskou adresou spjatá další položka nastavení, kterou je volba autentizace (podrobněji v kapitole 3.2) a úroveň zabezpečení (kapitola 3.4).

Základní náležitosti klientských adres:

- Délka adresy by vždy měla být velikosti 1 bajt
- Určují, na kterou sadu objektů bude mít klient přístup
- Spojeno s úrovní autentizace

Je specifikováno několik hodnot, které určují využití, avšak ne všichni výrobci je dodržují [9]. Rezervované hodnoty jsou uvedeny v následující tabulce 2.1.

2.1.2 Server Address

`Server Address`, lze se také setkat s označením `ServerID` nebo `Server_SAP`, je adresa, pomocí které specifikujeme, na které zařízení se chceme připojit. Více meterů může být připojeno k jednomu komunikačnímu kanálu, případně metery mohou být za sebou zřetězené, a tak je nutné tuto adresu správně použít.

Server adresa se skládá z více částí, kde jednotlivé části jsou závislé na použitém rozhraní (HDLC x Wrapper viz kapitola 4.1).

Tab. 2.1: Tabulka používaných klientských adres [5]

Účel, označení	**Šestnáctkově**	**Dekadicky**
Nelze použít	0x00	0
Správa klienta	0x01	1
Veřejný klient	0x10	16
Pro volné použití	0x02 - 0x0F	2 - 15
Pro volné použití	0x11 - 0xFF	17 - 255
Low*	0x11	17
High*	0x12	18

*Běžně používaná kombinace adresy a úrovně autentizace.

Typy server adres se označují následovně:

- Serial Number
  - Skládá se pouze z jedné části
  - Shodné pro HDLC i Wrapper
  - Blíže popsáno v kapitole 2.1.3
- Default
  - U HDLC se skládá ze dvou částí (logická a fyzická adresa), kapitola 2.1.4
  - U Wrapperu se využívá pouze jedna část, označovaná jako `Physical Device`, tato část je shodná s fyzickou adresou u HDLC
- Custom
  - Shodné s předchozím, lze také použít pro vlastní výpočet ze sériového čísla, jelikož zde nedochází k žádným přepočtům

2.1.3 Adresace s využitím sériového čísla

Pro výpočet adresy ze sériového čísla se využívá jednoduchý vzorec. Výsledná adresa bude **2 bajty** dlouhá a použité vstupní sériové číslo může být jakkoli dlouhé [5]. Základní vzorec pro výpočet vypadá následovně:

$$\mathrm{SN_o} = (\mathrm{SN} \bmod 10\,000) + 1\,000, \tag{2.1}$$

kde SN je vstupní sériové číslo.

K výsledné hodnotě se ještě následně přidá hodnota 0x4000 pomocí funkce OR[1].

Pro lepší pochopení problematiky výpočtu adresy ze sériového čísla si ukážeme názorný příklad s použitím vstupního sériového čísla SN = 735 853 129. V následujícím příkladu je konečná hodnota převedena až do šestnáctkové soustavy, jelikož

[1]K přidání hodnoty by mělo dojít i v případě použití vlastního vzorce.

u přenosu s použitím Wrapperu se v hlavičce bude vyskytovat právě tato hodnota. Při použití HDLC se výsledná hodnota ještě dále upravuje. Proces úpravy na HDLC adresu je popsán v kapitole 2.1.4.

$$\begin{aligned} \mathrm{SN_o} &= (735\ 853\ 129 \bmod 10\ 000) + 1\ 000, \\ \mathrm{SN_o} &= 3\ 129 + 1\ 000, \\ \mathrm{SN_o} &= 4\ 129, \end{aligned} \tag{2.2}$$

následně provedeme převod do binární soustavy, kde je funkce OR lépe patrná,

$$\begin{aligned} 4\ 129 &\rightarrow 0001\ 0000\ 0010\ 0001_B, \\ &\text{OR} \\ \text{0x4000} &\rightarrow 0100\ 0000\ 0000\ 0000_B, \\ SN_o = 0101\ 0000\ 0010\ 0001_B &\rightarrow 20\ 513 \rightarrow \text{0x5021} \end{aligned}$$

Výsledná hodnota adresy ze sériového čísla je tedy rovna 20 513, případně 0x5021 v šestnáctkové soustavě. Tato hodnota by byla snadno patrná přímo ve Wrapper rámci, ale v HDLC rámci by ještě došlo k úpravám, které jsou popsané v následující kapitole.

2.1.4 Adresace při použití HDLC

Jak již bylo zmíněno, adresace při použití HDLC je o něco složitější, než při použití Wrapperu. Pravidla pro převod výsledných adres se vztahují na klientskou i serverovou adresu. Server adresa se vytváří ze sériového čísla nebo pomocí kombinace logické a fyzické adresy, které se jinak také označují jako upper a lower HDLC adresy (přehledně v tabulce 2.2). Výsledná délka musí být jeden, dva nebo čtyři bajty dlouhá, možný formát je uveden na obrázku 2.1. [5]

Tab. 2.2: Značení částí server adresy

Klasické značení	**HDLC označení**	**Alternativní označení**
Logická adresa	Upper HDLC address	Logical server
Fyzická adresa	Lower HDLC address	Physical server

Spojení logické a fyzické adresy záleží na jejich hodnotách následovně [5]:

- Hodnota obou adres je menší než 128 a zároveň požadovaná velikost adres je menší než 4 bajty
  - V tomto případě se logická adresa posune o 7 bitů vlevo a následně dojde ke spojení funkcí OR s fyzickou adresou

- Hodnota obou adres je menší než 16 384
  - V tomto případě se logická adresa posune o 14 bitů vlevo a následně se spojí funkcí OR s fyzickou adresou

Jakmile máme získané obě adresy (client i server), lze tyto adresy transformovat do HDLC formátu. Zde opět záleží na hodnotě a požadované velikosti adresy [5]:

- Pokud je hodnota adresy menší než 128 a požadovaná velikost je menší než 2 bajty, použije se vzorec 2.3
- Pokud je hodnota adresy menší než 16 384 a požadovaná velikost je menší než 4 bajty, použije se vzorec 2.4
- Pokud je hodnota adresy menší než 268 435 456, použije se vzorec 2.5

$$\mathrm{A}_1 = H << 1 \vee \ 1, \tag{2.3}$$

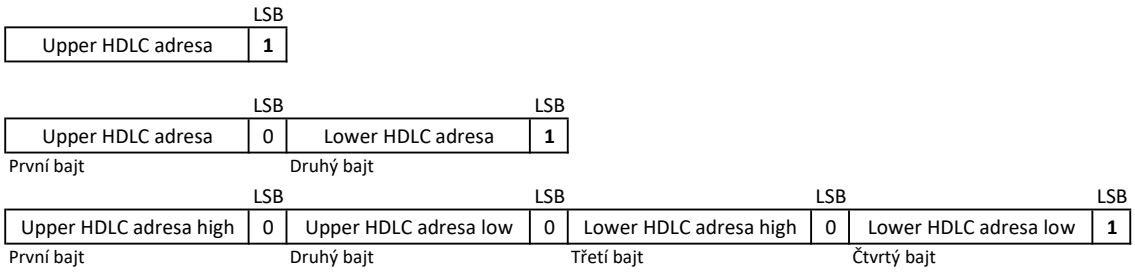
$$\begin{aligned} \mathrm{A}_2 = &(H \wedge 0\mathrm{x}3F80) \ << \ 2 \vee \\ &(H \wedge 0\mathrm{x}7F) \ << 1 \ \vee \ 1, \end{aligned} \tag{2.4}$$

$$\begin{aligned} \mathrm{A}_3 = &(H \wedge 0\mathrm{x}FE00000) \ << 4 \vee \\ &(H \wedge 0\mathrm{x}1FC000) \ << \ 3 \vee \\ &(H \wedge 0\mathrm{x}3F80) \ << \ 2 \vee \\ &(H \wedge 0\mathrm{x}7F) \ << \ 1 \ \vee \ 1, \end{aligned} \tag{2.5}$$

kde H představuje adresu, $\vee$ funkci OR, $\wedge$ funkci AND a $<<$ bitový posun vlevo.

Výpočty je možné zjednodušit, pokud se neprovede spojení fyzické a logické adresy. Předchozí výpočet je složitější, jelikož musí brát v úvahu formát HDLC adresy, který vyžaduje, že každý bajt je zakončen bitem 0. Pouze v posledním bajtu adresy musí být poslední bit (LSB) nastaven na 1. Tím je možné určit, kdy nastane konec adresy a může následovat další pole, formát HDLC je popsán v kapitole 4.1. [5]

Pro server adresu jsou validní pouze zápisy, které jsou uvedeny na obrázku 2.1



Obr. 2.1: Struktura server adresy u HDLC [5]

Na následujícím příkladu je zápis vyobrazen lépe a lze podle něj snadněji provést převod na výslednou adresu. Velikost server adresy jsou 4 bajty, jedná se tak o nejdelší možný zápis. Hodnoty na příkladu jsou následující:

$$\begin{aligned} \text{Upper HDLC adresa} = 4\ 660 &\rightarrow \text{0x1234} \rightarrow 0001\ 0010\ 0011\ 0100_B \\ \text{Lower HDLC adresa} = 16\ 383 &\rightarrow \text{0x3}FFF \rightarrow 0011\ 1111\ 1111\ 1111_B \\ \text{Klientská HDLC adresa} = 58 &\rightarrow \text{0x3}A \rightarrow 0011\ 1010_B \end{aligned}$$

Zvolené adresy jsou rozděleny do jednotlivých bajtů. Celé výsledné pole je vyobrazeno v následující tabulce 2.3

Tab. 2.3: Vzorový příklad adresy [5]

Serverová adresa				Klientská adresa
Logická adresa		Fyzická adresa		
Upper HDLC high	Upper HDLC low	Lower HDLC high	Lower HDLC low	HDLC adresa
4660 -> **01 0010 0**011 0100_B		16 383 -> **11 1111 1**111 1111_B		58 -> 011 1010_B
0 1 0 0 1 0 0 **0**	0 1 1 0 1 0 0 **0**	1 1 1 1 1 1 1 **0**	1 1 1 1 1 1 1 **1**	0 1 1 1 0 1 0 **1**
První bajt	Druhý bajt	Třetí bajt	Čtvrtý bajt	Pátý bajt
0x**48**	0x**68**	0x**FE**	0x**FF**	0x**75**
Cílová adresa				Zdrojová adresa

V samotném HDLC rámci by tedy adresa byla zobrazena jako **48 68 FE FF 75** (v šestnáctkovém zápisu) ve směru od klienta k serveru. V opačném směru by zápis byl následující **75 48 68 FE FF**.

3 Zabezpečení DLMS

Standard DLMS definuje použití řady zabezpečovacích mechanizmů a nespoléhá pouze na zabezpečení samotného spojení (například tunelování pomocí IPsec). Mezi zabezpečovací mechanizmy patří identifikace, autentizace, autorizace a šifrování.

V následující kapitole jsou popsány využívané mechanismy, které jsou definované ve standardu. Avšak samotný standard neomezuje výrobce, aby si implementovali vlastní řešení zabezpečení. Bohužel v takovém případě již nemusí být zajištěna interoperabilita.

3.1 Průběh zabezpečení

Nejdůležitějším krokem pro samotné zabezpečení je vytvoření spojení. V tomto kroku dochází k identifikaci zařízení, případně i k identifikaci uživatelů. Dále dochází k autentizaci stran a případně k dohodnutí na šifrovaném spojení.

Vytvoření spojení se u DLMS označuje jako AA (Application Associations) [5]. V této fázi dochází k dohodnutí podporovaných služeb, konfiguraci spojení (Application Context), určení úrovně zabezpečení a k výměně informací potřebných pro samotné zabezpečení. Průběh navázání spojení (AA) s různými úrovněmi zabezpečení je vyobrazen na obrázku 3.1. Úrovně odpovídají úrovním zabezpečení, které jsou popsány v následující části.

3.2 Autentizace

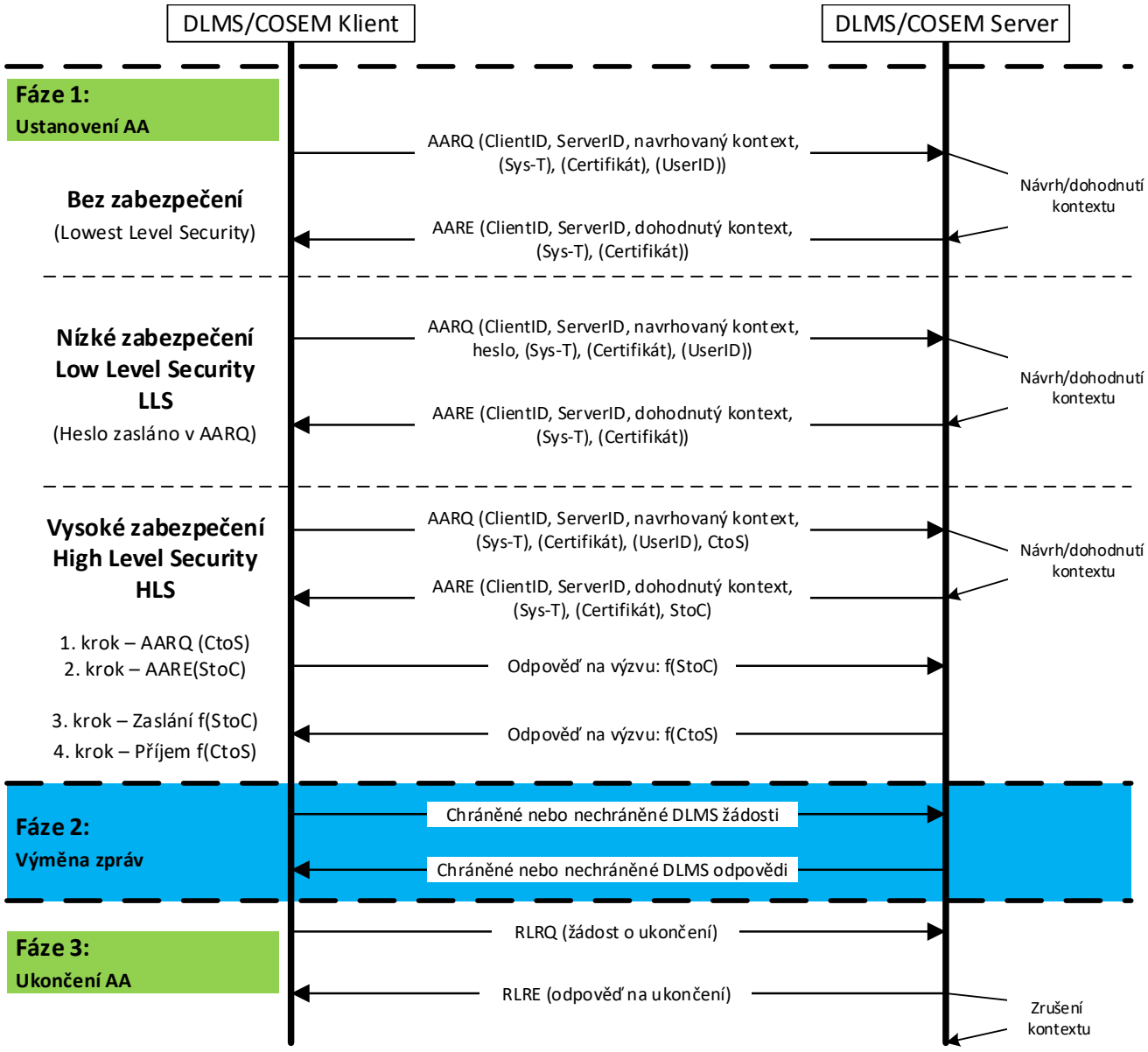
Autentizace patří v informační bezpečnosti mezi jednu z nejdůležitějších služeb bezpečnosti. Hlavním cílem autentizace je ověření totožnosti entit (komunikujících stran). V DLMS se autentizace provádí v průběhu navazování spojení. Ověřovat lze jak klientskou stranu, tak i stranu serveru (tedy chytrý meter). Případně lze také ověřit identitu uživatele na klientské straně. Průběh autentizace je znázorněn na obrázku 3.1

Existují tři druhy autentizace v rámci specifikace DLMS [5]:

- Lowest Level Security
  - Bez autentizace
- Low Level Security (LLS)
  - Autentizace pomocí sdíleného hesla. Heslo je zasíláno v čitelné podobě
- High Level Security (HLS)
  - Nejvyšší stupeň autentizace v DLMS
  - Založeno na modelu výzva-odpověď

– Navázání spojení je složitější (čtyřcestné)
– Dochází k modifikacím poskytnutých výzev za pomoci bezpečnostních mechanizmů a společného klíče

V rámci fáze navázání spojení (AA) dochází k zaslání položky označované jako `authentication_mechanism_name` nesoucí hodnotu `mechanism_id` [5]. Tato hodnota přesně určuje, jaký typ autentizace se vyžaduje. V tabulce 3.1 jsou uvedeny všechny aktuální možnosti autentizace.



Obr. 3.1: Průběh navázání spojení [5]

Tab. 3.1: Tabulka autentizačních mechanismů [5]

Druh autentizace	**Mechanism_id**
Bez zabezpečení	0
LLS	1
HLS*	2
HLS s MD5	3
HLS s SHA-1	4
HLS s GMAC	5
HLS s SHA-256	6
HLS s EC-DSA	7

*Možné proprietární řešení

3.2.1 No security (Lowest Level Security)

Tato úroveň autentizace se také označuje jako „bez zabezpečení“. Primárním účelem je zjištění základních informací ze serveru [5]. Klient tak může získat přístup pouze k některým COSEM objektům a atributům, které má server specifikován v rámci `Security context` a dále pomocí přístupových práv jednotlivých atributů.

3.2.2 Low Level Security

V rámci této úrovně autentizace se klient ověřuje serveru pomocí hesla, které se zasílá v otevřené podobě [5]. K validnímu ověření musí server stejné heslo znát. Heslo by na straně serveru mělo být uloženo v rámci objektu `Association SN/LN`.

V případě shodného hesla je fáze AA dokončena a klient má přístup k objektům na serveru. V opačném případě server zašle chybovou hlášku a dojde k uzavření spojení.

3.2.3 High Level Security

Úroveň HLS je aktuálně nejvyšším autentizačním zabezpečením, které DLMS poskytuje. Pro úspěšné navázání spojení se musí klient a server vzájemně autentizovat. Celý průběh navazování spojení je v tomto případě čtyřcestný [5] a je znázorněný na obrázku 3.1.

Průběh AA lze popsat následovně ve 4 krocích [5]:

1. V žádosti AARQ dochází k zaslání výzvy od klienta k serveru – *CtoS* + dodatečné informace.
2. V odpovědi AARE je zaslána výzva od serveru pro klienta – *StoC* + dodatečné informace.

- Pokud by nastala situace, že *CtoS* = *StoC*, musí dojít k odmítnutí žádosti a k ukončení spojení.

3. Klient zpracuje *StoC* a dodatečné informace a odešle výsledek serveru uvnitř `Action Request` zprávy. Po přijetí provede server stejný výpočet a porovná výsledky. V případě shody dojde k úspěšné autentizaci klienta k serveru a lze přistoupit na 4. krok.
4. Server zpracuje *CtoS* a dodatečné informace a odešle výsledek v rámci zprávy `Action Response`. Klient provede po přijetí zprávy kontrolu s vlastním výsledkem. V případě shody dojde k úspěšnému ověření serveru a spojení je tím úspěšně navázáno na obou stranách

Pokud nebudou údaje souhlasit ve 3. nebo 4. kroku, tak dojde k odeslání chybové hlášky a následně je spojení ukončeno.

Po 2. kroku server udělí přístup pouze k metodám pro zaslání odpovědi na `Action Request` zprávu. Tato metoda se označuje jako `reply_to_HLS_authentication` a je obsažena v aktuální `Associaci LN/SN` [5].

Všechny kroky jsou podrobněji znázorněny v tabulce 3.2.

HLS podporuje následující mechanizmy pro autentizaci, jejich přidělené ID jsou uvedeny v tabulce 3.1:

- MD5
- SHA-1
- GMAC
- SHA-256
- EC-DSA
- Případně proprietární pro výrobce

3.3 Další součásti zabezpečení

V průběhu DLMS spojení se využívají i další dodatečné informace, jako jsou například přístupová práva k objektům, či nastavení zabezpečení, které bývá specifikováno uvnitř položky označované jako `Security context` [5].

3.3.1 Security context

Security context, česky lze označit jako kontext zabezpečení, je položka, která je spravována uvnitř objektu **Security setup** a obsahuje používané/vyžadované nastavení zabezpečení.

Obsahuje tyto položky:

- Požadovaný Security suite, více v kapitole 3.4.
- Bezpečnostní politika – způsob zabezpečení.
- Zabezpečovací materiály, jako jsou klíče, certifikáty, inicializační vektory atd.

Tab. 3.2: Průběh HLS autentizace [5]

Authentication mechanism	**Krok 1: C → S**	**Krok 2: S → C**	**Krok 3: C → S f(StoC)**	**Krok 4: S → C f(CtoS)**
	Neseno ve zprávě:			
	AARQ	**AARE**	**Action**[1]**.request** odpověď na *StoC*	**Action**[1]**.response** odpověď na *CtoS*
mechanism_id(2) HLS spec. výrobcem	*CtoS* : Náhodný string 8-64 oktetů	*StoC* : Náhodný string 8-64 oktetů	Spec. výrobcem	Spec. výrobcem
mechanism_id(3) HLS MD5			**MD5**(*StoC* ‖ HLS Secret)	**MD5**(*CtoS* ‖ HLS Secret)
mechanism_id(4) HLS SHA-1			**SHA-1**(*StoC* ‖ HLS Secret)	**SHA-1**(*CtoS* ‖ HLS Secret)
mechanism_id(5) HLS GMAC	*CtoS* : Náhodný string 8-64 oktetů Případně: Systém-Title-C	*StoC* : Náhodný string 8-64 oktetů Případně: Systém-Title-S	SC ‖ IC ‖ **GMAC** (SC ‖ AK ‖ *StoC*)	SC ‖ IC ‖ **GMAC** (SC ‖ AK ‖ *CtoS*)
mechanism_id(6) HLS SHA-256			**SHA-256** (HLS_Secret ‖ SystemTitle-C ‖ SystemTitle-S ‖ *StoC* ‖ CtoS)	**SHA-256** (HLS_Secret ‖ SystemTitle-S ‖ SystemTitle-C ‖ *CtoS* ‖ *StoC*)
mechanism_id(7) HLS ECDSA	*CtoS* : Náhodný string 8-64 oktetů Případně: Systém-Title-C Cert-Sign-Client	*StoC* : Náhodný string 8-64 oktetů Případně: Systém-Title-S Cert-Sign-Server	**ECDSA**(SystemTitle-C ‖ SystemTitle-S ‖ *StoC* ‖ *CtoS*)	**ECDSA**(SystemTitle-S ‖ SystemTitle-C ‖ *CtoS* ‖ *StoC*)
Legenda: - C: Klient, S: Server, CtoS: Výzva klienta serveru, StoC: Výzva serveru klientovi - IC: Invocation Counter, SC: Security Control byte, AK: Authentication key [1] Ve většině případů je použita zpráva typu Action				
Mechanizmy 3 a 4 je doporučené pro nové implementace již nevyužívat. Certifikáty a System-Titles je nutné zaslat pouze, pokud již nejsou známé druhé straně.				

3.3.2 Přístupová práva

Přístupová práva jsou specifikovaná v objektech `Asociace SN/LN` a jsou na nich závislá. Je možné je specifikovat 2 způsoby. První způsob je základní a obsahuje následující oprávnění [5]:

- Přístup k atributům
  - no_access
  - read_only
  - write_only
  - read_and_write
- Přístup k metodám
  - no_access
  - access

Druhý způsob se využívá v případě, že je nastavena vyšší úroveň bezpečnostní politiky. Pro definici přístupových práv se využívá 8 bitů. Hodnota každého bitu pak specifikuje následující oprávnění:

Tab. 3.3: Přístupová práva, význam bitů [5]

Bit	**Přístup k atributům**	**Přístup k metodám**
0	Čtení	Přístup
1	Zápis	-nevyužito-
2	Autentizovaná žádost	Autentizovaná žádost
3	Šifrovaná žádost	Šifrovaná žádost
4	Digitálně podepsaná žádost	Digitálně podepsaná žádost
5	Autentizovaná odpověď	Autentizovaná odpověď
6	Šifrovaná odpověď	Šifrovaná odpověď
7	Digitálně podepsaná odpověď	Digitálně podepsaná odpověď

3.4 Šifrování

Mezi důležité prvky informační bezpečnosti patří utajování dat, kterého je docíleno pomocí šifrovacích algoritmů. DLMS specifikuje hned několik možností, které je možné při spojení s meterem využít. Tyto možnosti se označují jako `Security suite` (česky lze přeložit jako „Zabezpečovací sada“, či zkráceně SS). Aktuálně jsou definovány 3 sady [5], kde každá sada má vlastní ID (hodnota 0-15). Jejich ID a jednotlivé algoritmy jsou vyobrazeny v následující tabulce 3.4.

3.4.1 Security suite 0

Jak již vyplývá z tabulky 3.4, tak `Security Suite 0` je nejzákladnějším stupněm zabezpečení důvěrnosti dat. Poskytuje pouze šifrování dat za pomoci standardu AES-128bit v GCM módu. K šifrování se využívají sdílená hesla, které jsou známá na obou stranách [5].

Tab. 3.4: Security suites [5]

ID	Označení	Šifrování	Digitální podpis	Dohodnutí klíče	Hash	Přenos klíče*	Komprese
0	AES-GCM-128	AES-GCM-128	-	-	-	AES-128	-
1	ECDH-ECDSA-AES-GCM-128-SHA-256	AES-GCM-128	ECDSA s P-256	ECDH s P-256	SHA-256	AES-128	V.44
2	ECDH-ECDSA-AES-GCM-256-SHA-384	AES-GCM-256	ECDSA s P-384	ECDH s P-384	SHA-384	AES-256	V.44
3-15	Rezervováno pro budoucí použití						

*Klíč je při přenosu zašifrován (zabalen)

3.4.2 Security suite 1 a 2

Z tabulky 3.4 vyplývá, že `Security suite 1 a 2` jsou si velmi podobné. Jediným rozdílem jsou použité velikosti klíčů.

Pro šifrování dat se využívá standard AES 128/256 bit v GCM módu. Pro dohodnutí klíče se využívá ECDH na eliptické křivce P-256 nebo P-384, kde se pomocí dohodnutého klíče zašifruje samotný šifrovací klíč, který se na druhou stranu zašle utajený pomocí AES-128/256 bit.

Zprávy jsou podepisovány pomocí asymetrického algoritmu ECDSA založeného na křivkách P-256 nebo P-384 [5].

3.4.3 Komprese

V tabulce 3.4 je v posledním sloupci zmíněna komprese. Tato komprese neobsahuje žádné kryptografické metody, ale může být využívána k transformaci a zmenšení xDLMS APDU [5]. Využívá se především u symetrických algoritmů. Kompresní algoritmus vychází z ITU-T V.44: 2000 [10] a měl by splňovat následující požadavky:

- nízké výpočetní zatížení,
- nízké nároky na paměť,
- nízká odezva.

3.4.4 Security Header

Každá zpráva, která podléhá zabezpečení obsahuje hlavičku s parametry, jaký typ zabezpečení se pro danou zprávu používá. Tato hlavička obsahuje tzv. `security control byte` [5], který se skládá z několika parametrů uvedených v tabulce 3.5. Kromě toho hlavička obsahuje čítač označovaný jako `invocation counter` dlouhý 32 bitů. Celá hlavička je tedy dlouhá 40 bitů, neboli 5 bajtů.

Tab. 3.5: Security Control Byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3-0
Komprese	Key_Set	E	A	Security Suite ID

- Bit 0-3: Číslo sady (zde je patrný max. počet sad = 16)
- Bit 4: „A“ značí využití autentizovaných zpráv
- Bit 5: „E“ značí využití šifrovaných zpráv
- Bit 6: Key_Set: 0 = Unicast, 1 = Broadcast
- Bit 7: Využití komprese

3.5 Použité algoritmy

Pro zabezpečení se v DLMS používá množství kryptografických algoritmů. Mezi hlavní patří následující algoritmy (s popisem týkajícím se DLMS) [5]:

- AES
  - Advanced Encryption Standard
  - Standard založený na symetrickém algoritmu Rijndael
  - Využitý především pro šifrovaný přenos DLMS dat
  - Také se používá pro přenos dohodnutého šifrovacího klíče
  - Použité 128 a 256 bitové verze
  - S využití GCM módu (Galois/Counter Mode) [11]
- EC-DH
  - Diffie-Hellmanův protokol na eliptických křivkách
  - Pro dohodnutí šifrovacího klíče
  - Využit na křivkách P-256 a P-384
- EC-DSA
  - Digital Signature Algorithm na eliptických křivkách
  - Asymetrický algoritmus pro elektronický podpis
  - Využit na křivkách P-256 a P-384
- SHA
  - Secure Hash Algorithm
  - Hašovací algoritmus (tvorba otisku zpráv)
  - Využité 256 a 384 bitové varianty
  - Využit pro digitální podpis, výměnu klíčů a také pro autentizaci
- V.44
  - Kompresní algoritmus
  - Může také sloužit ke kontrole chyb
  - Neobsahuje žádnou kryptografii

3.5.1 Parametry asymetrické kryptografie v DLMS

V DLMS se pro asymetrickou kryptografii využívají pouze eliptické křivky. Organizace NIST [12] doporučila v roce 2013 celkem 5 základních eliptických křivek, z nichž byly 2 vybrány pro použití v DLMS. Konkrétně se jedná o křivky uvedené v následující tabulce 3.6.

Tab. 3.6: Eliptické křivky používané v DLMS [5]

DLMS Security Suite	Křivka dle [12]	ASN.1 označení	Další označení
Suite 0	-	-	-
Suite 1	NIST P-256	1.2.840.10045.3.1.7	secp256r1
Suite 2	NIST P-384	1.3.132.0.34	secp384r1

3.5.2 Certifikáty

V DLMS jsou dle specifikace [5] vyžadovány pouze certifikáty typu X.509 ve verzi 3. Každý takový certifikát se skládá z 3 hlavních částí: údaje certifikátu, algoritmus podpisu a samotný podpis certifikátu od certifikační autority.

Údaje o certifikátu musí obsahovat následující položky:

- Verze certifikátu – v případě DLMS pouze v3
- Sériové číslo – číslo přidělené certifikační autoritou do max. délky 20 bajtů
- Vydavatel – základní údaje o vydavateli certifikátu
- Platnost od – do
- Subjekt – komu je certifikát vydán (viz níže)
- Údaje o veřejném klíči
- Rozšiřující údaje

Subjekt certifikátu musí obsahovat identifikátor zařízení, tzv. `System-title`. Identifikátor se následně zapisuje do atributu `Common Name`, zkráceně `CN` a musí být zapsán hexadecimálně v 8 bajtech. Příklad subjektu pro `System-title`: „ABCDEFGH“ bude v poli `subject` v certifikátu zapsáno následovně:

CN=4142434445464748

V rozšiřujících údajích se v DLMS nachází pole `KeyUsage`, které specifikuje, k jakému účelu lze daný certifikát použít. Základní použití je pro dohodnutí klíčů a dále pro digitální podpis.

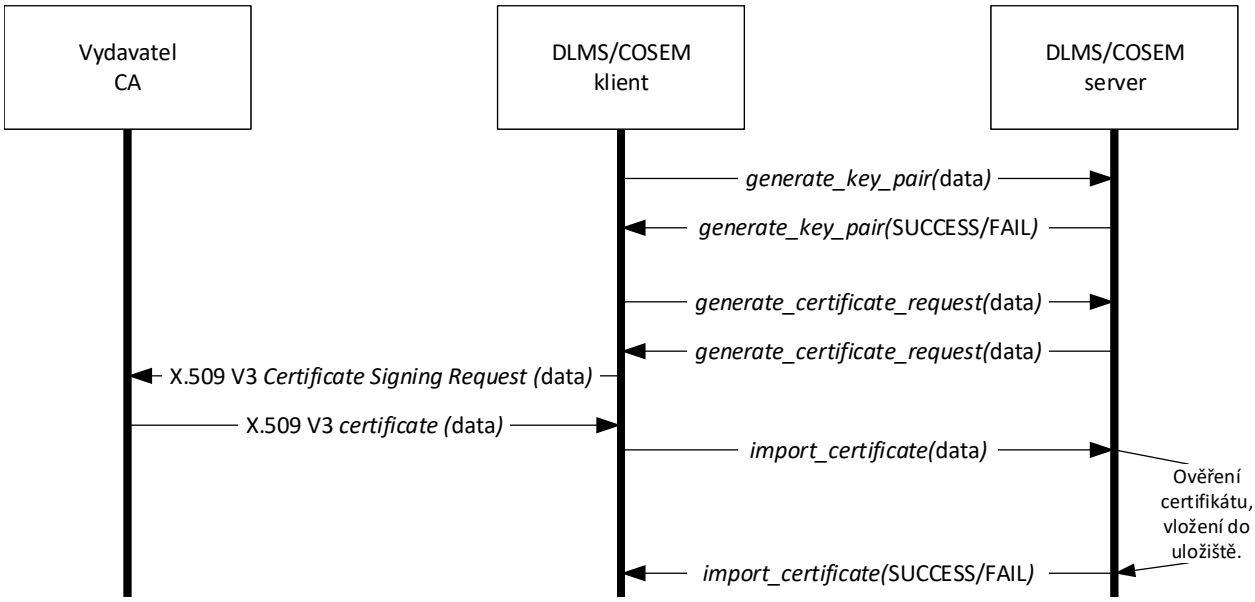
Při tvorbě serveru je možné se setkat s pojmem „personalizace zabezpečení“ (security personalisation) [5], kde tento pojem znamená poskytnutí serveru s asymetrickými páry klíčů korespondující certifikáty pro veřejné klíče. Tento proces je

možné provést dvěma základními způsoby. Prvním je poskytnutí těchto klíčů a certifikátů přímo výrobcem při výrobě elektroměru, kde však soukromé klíče musí zůstat bezpečně uložené přímo na serveru a nikdy nesmí být žádným způsobem odhalené. Druhou možností je využít objekt `Security Setup`, který obsahuje potřebné metody pro práci s certifikáty. Tento druhý proces může být využit jak při výrobě, tak i kdykoliv během životnosti elektroměru pro vygenerování nových klíčů a získání potřebných certifikátů.

Metody objektu `Security Setup` jsou následující [5]:

- Dohodnutí klíče (`key_agreement`)
- Vygenerování nového páru klíčů (`generate_key_pair`)
- Vytvoření žádosti o nový certifikát (`generate_certificate_request`)
- Import certifikátu (`import_certificate`)
- Export certifikátu (`export_certificate`)
- Odstranění certifikátu (`remove_certificate`)

Postup generování nových klíčů a certifikátů je znázorněn na obrázku 3.2.



Obr. 3.2: Tvorba certifikátu pro server [5]

Celý proces lze rozdělit do 4 kroků:

1. Klient vyvolá metodu `generate_key_pair` a v parametrech vybere o jaký typ[1] klíče se bude jednat. Server pouze odpoví, zdali generování bylo úspěšné.
2. Klient vyvolá metodu `generate_certificate_request`, kde parametry obsahují, pro který pár klíčů má být vytvořena žádost CSR (Certificate Signing

[1]Typ pro digitální podpis, dohodnutí klíčů nebo TLS

Request). Žádost je podepsána soukromým klíčem z párů, kterého se daná žádost týká. Server odešle zprávu obsahující CSR.

3. Klient zašle žádost o certifikát (CSR) certifikační autoritě, která zkontroluje, zdali žádost splňuje všechny požadavky a následně vytvoří a vydá certifikát, který zašle zpátky klientovi.
4. Klient vyvolá metodu `import_certificate`, kde parametr obsahuje certifikát vydaný certifikační autoritou. Server následně ověří certifikát a odešle oznámení, zda byl import úspěšný.

Server by vždy měl obsahovat pouze jeden pár klíčů pro dané účely [5]. V případě úspěšného importu certifikátu dojde ke změně původní dvojice klíčů za nově vygenerované. Současně s tímto by mělo dojít k odstranění původních klíčů a původního certifikátu.

3.6 Dohodnutí klíče

Pro využití šifrovaného spojení je důležité, aby se komunikující strany shodli na použitém symetrickém klíči pro šifrování. V případě security suite 0 se využívá předsdílený klíč, který se většinou využívá stále stejný i pro více spojení. V případě security suite 1 a 2 lze využít jedno ze tří schémat pro výměnu klíče. Tyto schémata pochází z doporučení NIST. Konkrétně se jedná o NIST SP 800-56A Rev. 2: 2013 [13], avšak na tuto publikaci byla vydána další revize NIST SP 800-56A Rev. 3: 2018 [14].

Vybraná schémata pro dohodnutí klíče v DLMS jsou následující [5]:

1. Ephemeral Unified Model C(2e, 0s, ECC CDH)
2. One-Pass Diffie-Hellman C(1e, 1s, ECC CDH)
3. Static Unified Model C(0e, 2s, ECC CDH)

Jednotlivá schémata se z publikace [14] označují „C(0e, 0s, xxx)“, kde číslo před „e“ značí počet ephemeral klíčů (klíče na jedno použití), dále číslo před „s“ představuje počet statických klíčů, které jsou v tomto případě uložené v certifikátech. Jako poslední je v závorce uveden algoritmus, který je pro výměnu využit. V případě DLMS se vždy jedná o Diffie Hellmanův protokol na eliptických křivkách.

4 Analýza DLMS zpráv

Pro účely testování bylo potřeba zjistit, jaké objemy dat se v rámci DLMS zasílají. Z tohoto důvodu bylo nutné zjistit, jak se jednotlivé zprávy vytvářejí.

Zprávy zasílané v rámci protokolu DLMS lze vytvářet dvěma způsoby. Jeden typ je zaměřen především do TCP/IP sítí, kdežto druhý se využívá zejména u sériových linek a podobných spojení. Označují se následovně [5]:

1. Wrapper
   - Vhodné v kombinaci s TCP
   - Nezajišťuje integritu dat, spoléhá na nižší vrstvy
   - Ve většině případů jsou zprávy kratší v porovnání se stejnou zprávou u HDLC. Především díky absenci kontrolních součtů.
2. HDLC
   - Vhodné pro sériové linky
   - Zajišťuje integritu pomocí až dvou kontrolních součtů
   - V rámci jednoho spojení vyžaduje zaslání více zpráv než wrapper

4.1 Složení PDU

V následující části budou znázorněny používané formáty jednotlivých typů zpráv. Ukázky budou provedeny na stejném obsahu dat, díky čemuž je možné provést jejich vzájemné porovnání.

4.1.1 Formát Wrapper

Formát zpráv typu wrapper je značně jednodušší než formát HDLC. Zprávy s použitím wrapperu se označují jako WPDU (Wrapper protocol data unit) [5].

Tab. 4.1: Formát Wrapper (verze 0x0001) [5]

Hlavička				Datová část
Verze	Zdrojová adresa	Cílová adresa	Délka	Data (APDU)

Každá část hlavičky se skládá z přesně 2 bajtů. Celá WPDU hlavička je tedy dlouhá 8 bajtů. Jednotlivé části WPDU jsou následující:

1. Verze – verze WPDU, aktuální hodnota je 00 01 a tato hodnota je spravována organizací DLMS UA [4].
2. Zdrojová adresa
3. Cílová adresa

4. Délka – délka dat v následujícím datovém poli
5. Data – nesená DLMS data označována jako APDU

4.1.2 Formát HDLC

Aktuálně se ve většině HDLC zpráv využívá následující formátování. Vychází z formátu typu 3 definovaném v Annexu H.4 obsaženého v normě IEC 13239 [5, 15].

Tab. 4.2: Formát HDLC zprávy [5]

Flag	Frame format	Cílová adresa	Zdrojová adresa	Control	HCS	Data	FCS	Flag

Jak si lze povšimnout, tak se HDLC formát skládá z 9 části, které jsou popsány níže:

1. Flag – počátek zprávy pomocí bajtu **`7E`**
2. Frame format – obsahuje typ formátu a dále popisuje velikost dat
3. Cílová adresa
4. Zdrojová adresa
5. Control
   - Obsahuje hodnotu tzv. frame counteru
   - Napomáhá k zajištění správného pořadí zpráv
   - V odpovědi automaticky dojde k přičtení 16
   - Následující žádost přičítá 1
6. HCS – kontrolní součet hlavičky HDLC
7. Data* – APDU
8. FCS – kontrolní součet celého rámce
9. Flag – konec zprávy pomocí bajtu **`7E`**

*Před samotná data se u HDLC ještě přidává tzv. LLC (jedná se o hodnoty E6 E6 00 a E6 E7 00), kde E6 bývá v žádosti od klienta a E7 v odpovědi serveru.

4.2 Porovnání HDLC a Wrapper zprávy

Pro ukázku si vezmeme zprávu, která je odpovědí na čtení z datového objektu `1.1.1.4.0.255` atribut 4. Jedná se o objekt typu `DemandRegister` – Sum Li Active power+ (QI + QIV) Current, kde aribut 4 udává jednotku.

Pro HDLC vypadá odpověď následovně:
7E A0 15 41 03 74 24 CB E6 E7 00 0C 01 00 02 02 0F 00 16 1B 61 63 7E

Pro Wrapper identická odpověď vypadá takto:
00 01 00 20 00 01 00 09 0C 01 00 02 02 0F 00 16 1B

S použitím DLMS překladače můžeme získat čitelnější zprávu. Na výpisku 4.1 je přeložena HDLC zpráva a na následujícím výpisku 4.2 je přeložená Wrapper zpráva. Zeleně označená část je společná pro oba formáty, jedná se o datovou část (APDU).

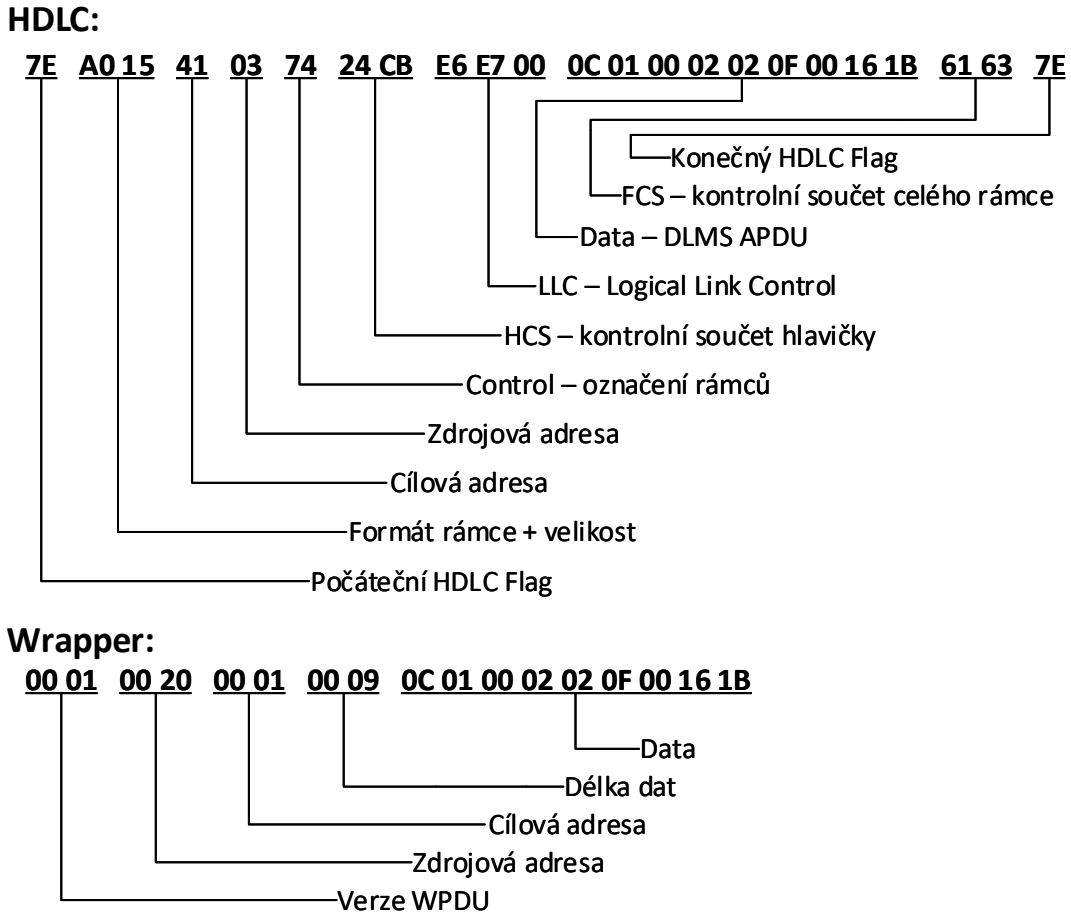
Výpis 4.1: Zobrazení HDLC zprávy za pomocí DLMS překladače

```
<HDLC len="20" >
  <TargetAddress Value="32" />
  <SourceAddress Value="1" />
  <!--I frame.-->
  <FrameType Value="74" />
  <PDU>
    <ReadResponse Qty="1" >
      <Data>
        <Structure Qty="2" >
          <Int8 Value="0" />
          <Enum Value="27" />
        </Structure>
      </Data>
    </ReadResponse>
  </PDU>
</HDLC>
```

Výpis 4.2: Zobrazení Wrapper zprávy za pomocí DLMS překladače

```
<WRAPPER len="17" >
  <TargetAddress Value="32" />
  <SourceAddress Value="1" />
  <PDU>
    <ReadResponse Qty="1" >
      <Data>
        <Structure Qty="2" >
          <Int8 Value="0" />
          <Enum Value="27" />
        </Structure>
      </Data>
    </ReadResponse>
  </PDU>
</WRAPPER>
```

Podle popsaných formátů z předchozích podkapitol 4.1.1, 4.1.2 již lze rozdělit zprávy na jednotlivé části, ze kterých se zprávy skládají 4.1:



Obr. 4.1: Analýza HDLC a Wrapper zprávy

Z formátů si lze povšimnout, že HDLC a Wrapper mají zapsané v opačném pořadí cílovou a zdrojovou adresu.

Dále u formátu wrapper je možné použít přímo bajty adresy, kde stačí provést převod z šestnáctkové do desítkové soustavy. Takže adresa odpovídající v šestnáctkové 00 20 lze snadno převést na výsledné číslo 32 dekadických.

Naopak u HDLC tento převod není až tak snadný, jelikož adresa může být zapsána několika způsoby, které závisejí na hodnotě adresy. Výpočet adres u HDLC je popsán v kapitole 2.1.4.

4.3 Překlad datové části

Příklad datové části 0C 01 00 02 02 0F 00 16 1B.

Většina datových částí vždy začíná jedním bajtem, který označuje o jaký příkaz se jedná. V příkladu se jedná o bajt **0C**, která znamená `Read_response`, čili odpověď na předchozí žádost o čtení.

Následuje bajt[1], který v případě odpovědi říká, kolik odpovědí zpráva obsahuje, v tomto případě se jedná o **01**. Pokud by se jednalo o žádost, tak by tento bajt obsahoval jednu ze 3 možností:

Tab. 4.3: Specifikace příkazu **0C** v žádosti

Typ příkazu	**Hodnota**
Normal	01
Next data block	02
With list	03

Následující bajt ukončuje příkaz a označuje začátek nesených dat. Jedná se o bajt **00**. Následují samotná nesená data, která jsou pro větší přehlednost rozdělena do tabulky:

Tab. 4.4: Rozdělení dat v APDU

Bajt	**Význam**
02	Nesená data jsou typu struktury (komplexní datový typ)
02	Struktura obsahuje 2 položky
0F	1. položka je hodnota int8 (velikost 1 bajt)
00	Hodnota int8 = 0
16	2. položka obsahuje typ Enum (velikost 1 bajt)
1B	Hodnota Enumu = 27

[1]Tento bajt může úplně chybět, případně hodnoty mohou mít jiný význam. Vše je závislé na příkazu z předchozího bajtu.

5 Implementace zabezpečení

Pro implementaci zabezpečení je použit programovací jazyk Java s DLMS knihovnou Gurux 5.2. Celé řešení je zakomponováno do vytvořené aplikace na testování DLMS, VUT DLMS Tester viz 5.1.

5.1 VUT DLMS Tester

Pro bakalářskou práci [8] byl vytvořen program pro základní testování DLMS. Tento program se stal základem pro vývoj rozsáhlejší testovací aplikace. Hlavní změnou oproti původnímu programu je především možnost definovat parametry pro více elektroměrů a následně s nimi komunikovat. Veškeré nastavení bylo přesunuto do uživatelského rozhraní, kde jej lze nyní snadno upravovat. Původně bylo možné program používat pouze na testované elektroměry v laboratoři.

Další změnou je ukládání vytvořených konfigurací a tak lze aplikaci opakovaně používat bez starostí se zadáváním nových parametrů. Aplikace také vytváří logovací soubory, které obsahují veškeré důležité dění v průběhu testování. Výstupní logy je pak možné analyzovat a provést celkové vyhodnocení testů, jako je například počet rozpadů spojení, průměrné délky odezvy, chybové zprávy od elektroměru, či chybně formátované zprávy.

Spolu s aplikací pro testování a odečítání (klient) došlo k vývoji i vlastní strany elektroměru (server), který umožňuje vytvoření požadovaného počtu elektroměrů na jednom zařízení. Kromě tvorby čistě vlastních konfigurací lze provést stažení konfigurace z reálného elektroměru a s výsledným souborem je možné provést jeho emulaci. Obě aplikace jsou díky použití jazyka Java nezávislé na operačním systému a splňují tak cíl práce pro implementaci na Raspberry Pi (RPi) viz kapitola 5.1.8.

V průběhu vývoje, byla aplikace testována s reálnými elektroměry různých výrobců, jako jsou ZPA, Landis+Gyr a Itron.

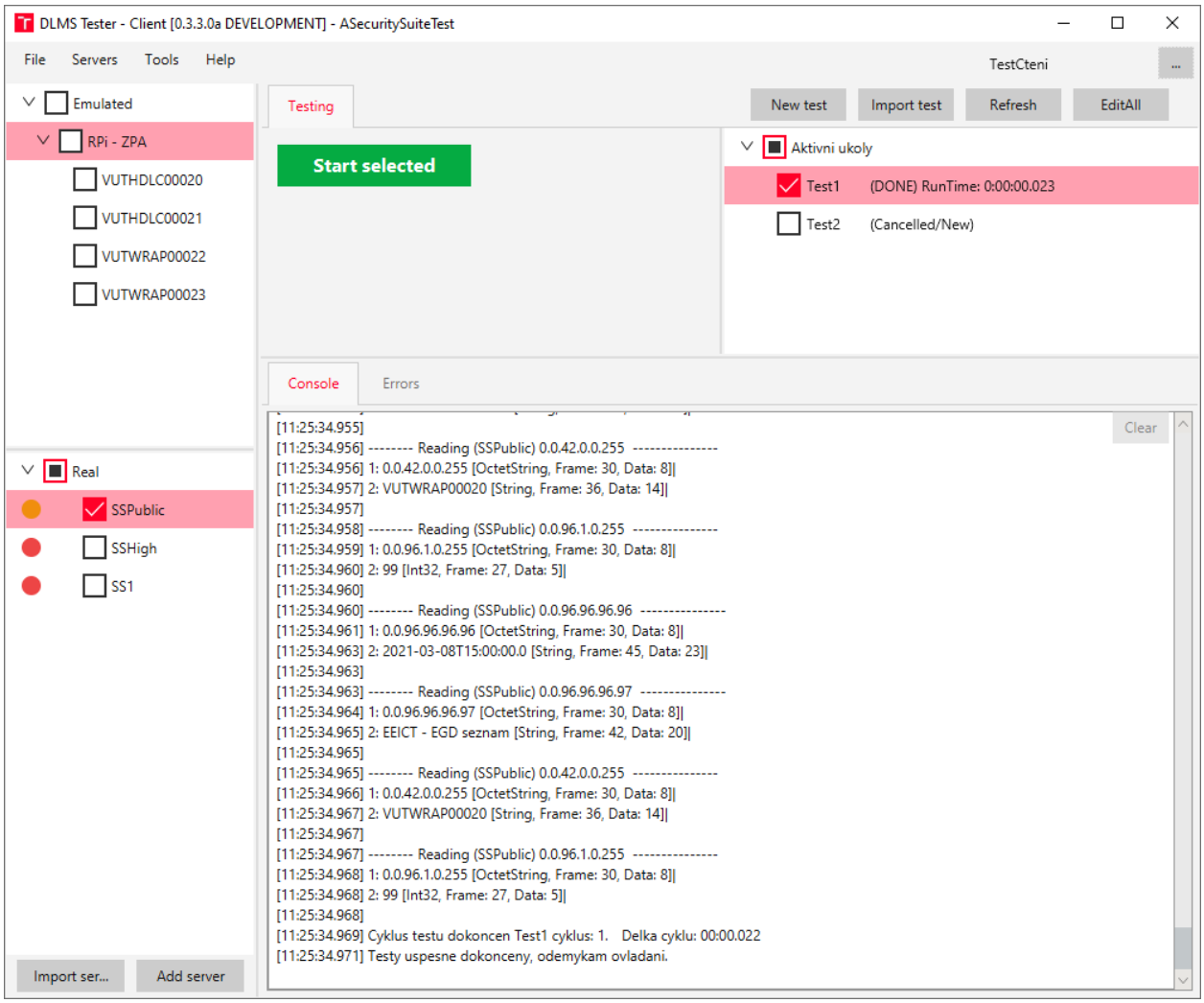
Na následujících stránkách jsou rozebrány klíčové části aplikace, které byly v rámci diplomové práce realizovány. Vždy se jedná o obrázek rozhraní a popis funkčnosti.

V případě zájmu je testovací aplikace dostupná u autora práce.

5.1.1 Hlavní okno aplikace

Na následujícím obrázku 5.1 je zachyceno hlavní okno aplikace. V dolní levé části se nachází přidané reálné elektroměry a naopak v horní části se nachází emulované elektroměry na konkrétním zařízení. Prostřední část je tvořena konzolovým výstupem, kam jsou směrovány informace o aktuálním průběhu testů. Nakonec je v horní části aktuálně pouze záložka testing, která obsahuje vytvořené testy a zobrazuje informace o jejich aktuálním průběhu[1]. V konzolovém poli je výpis čtení z jednoho proběhlého testu vyčteného z emulovaného elektroměru na RPi.

Pokud byla před vypnutím aplikace otevřená nebo uložená konfigurace, tak se aplikace při novém spuštění pokusí tuto konfiguraci načíst. Společně s předchozím nastavením si aplikace zapamatuje nastavený vzhled (normální, moderní světlý a moderní tmavý), doplňkovou barvu, rozložení okna, atd.



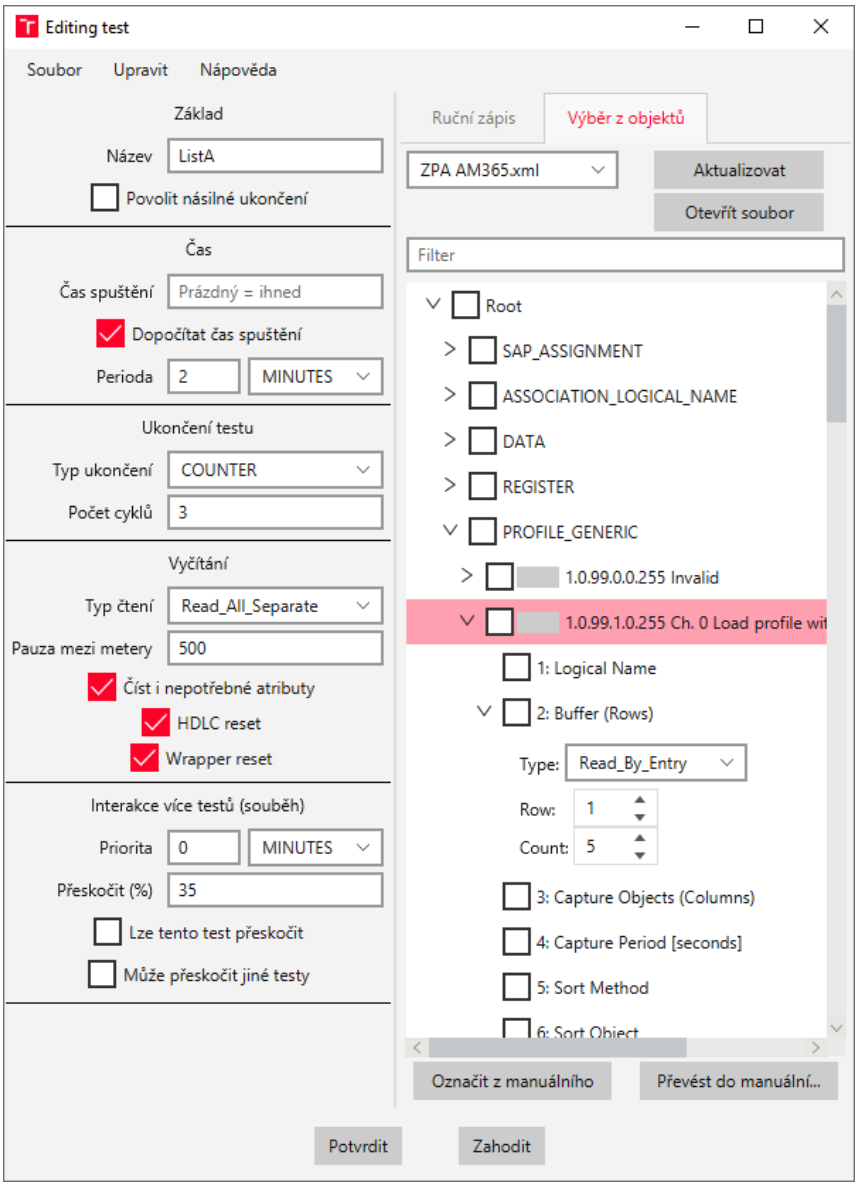
Obr. 5.1: Hlavní okno aplikace

[1] Aktuálně běžící test, čas do konce, počet cyklů, stav a čas do příštího spuštění

5.1.2 Konfigurace testů

Okno pro konfiguraci testů je znázorněno na obrázku 5.2. Toto nastavení lze rozdělit na dvě hlavní části. Levá část obsahuje nastavení času spuštění testu, periodu pro opakování cyklu a způsob ukončení daného testu. Ukončení je možné nastavit na určitý datum a čas, dále na celkovou dobu běhu a případně na počet provedených cyklů. Po nastavení všech časů lze specifikovat, zdali ke čtení využít možnost čtení pomocí listu, či každý atribut zvlášť. Nakonec je možné nastavit, zda vytvořené spojení při konci cyklu ukončit, nebo jej ponechat otevřené.

V pravé části lze nastavit objekty, které se budou testem vyčítat. Objekty je možné zapsat ručně nebo je možné je vybrat ze seznamu objektů na daném elektroměru. Kromě objektů je možné specifikovat i atributy a u historických objektů (`Profile Generic`) i způsob čtení záznamů (podle řádků nebo podle data a času).

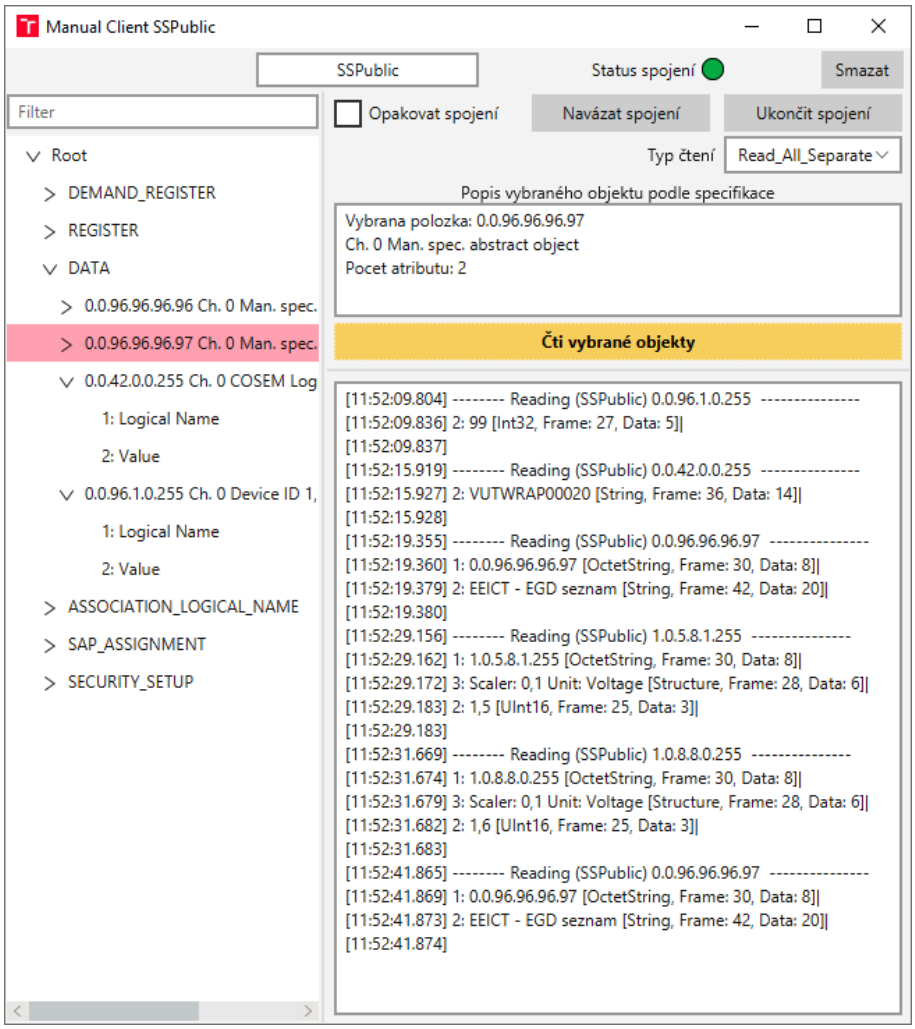


Obr. 5.2: Nastavení testů

5.1.3 Manuální čtení elektroměru

Pro kontrolu spojení nebo pro kontrolu obsahu některých objektů lze využít okno pro manuální vyčítání elektroměru zachycené na obrázku 5.3. Toto okno je velice jednoduché. V levé části si lze vybrat i několik objektů či atributů ke čtení. Tento seznam je téměř identický s výběrem objektů pro test. Po výběru objektů lze kliknout na tlačítko „Čti vybrané objekty“ a dojde k navázání spojení a vyčtení těchto objektů. Dále lze specifikovat, zda se má spojení po dokončení čtení ukončit, případně lze ručně navázat i ukončit spojení.

Poslední možnost v tomto okně je typ čtení pomocí listu, či každý atribut zvlášť. V okně se dále nachází 2 textová pole, kde první pole zobrazuje informace o vybraném objektu a v případě většího výběru se zde vypisuje velikost tohoto výběru. Poslední textové pole je pouze konzolový výstup, kam se zapisují údaje o čtených objektech a samotná vyčtená data. Konzolový výpis obsahuje čtení objektů z emulovaného elektroměru spuštěného ve vlastní serverové aplikaci na RPi.



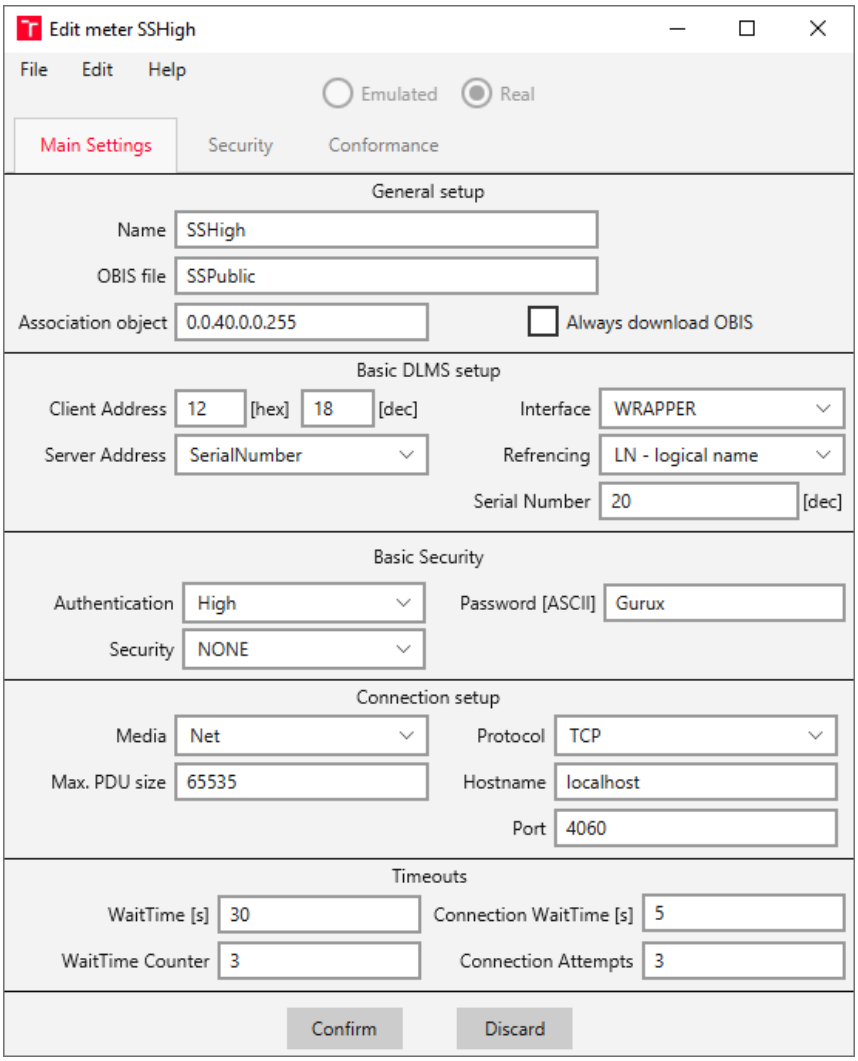
Obr. 5.3: Ruční vyčítání emulovaného elektroměru na RPi

5.1.4 Konfigurace připojení k elektroměru

Pomocí okna na obrázku 5.4 lze nakonfigurovat všechny parametry potřebné k správnému připojení k elektroměru. Celé nastavení je rozdělené do několika přehledných částí v jednotlivých rámečcích. V prvním se nachází pouze vlastní název pro zařízení a název souboru, kam se uloží asociace z elektroměru. Druhá část je velice důležitá pro připojení. Nachází se zde specifikování použitého rozhraní (viz 4.1.1 a 4.1.2), způsob odkazování a nastavení klientské (2.1.1) a serverové (2.1.2) adresy.

Následuje základní nastavení zabezpečení, které obsahuje pouze úroveň autentizace, heslo pro autentizaci a způsob zabezpečení zpráv. Pokročilé zabezpečení lze měnit v další záložce (viz kapitola 5.1.5) ve stejném okně.

Předposlední sekce obsahuje nastavení připojení k elektroměru. Na obrázku je použito běžné TCP/IP připojení pomocí IP adresy a portu. Kromě internetového média je možné aplikaci také použít s využitím sériové linky. Toho lze docílit výběrem „Serial“ v poli „Media“. Následně tak dojde ke změně požadovaných parametrů v pravé části sekce. Na konci se nachází časovače a počty pokusů pro obnovu spojení.

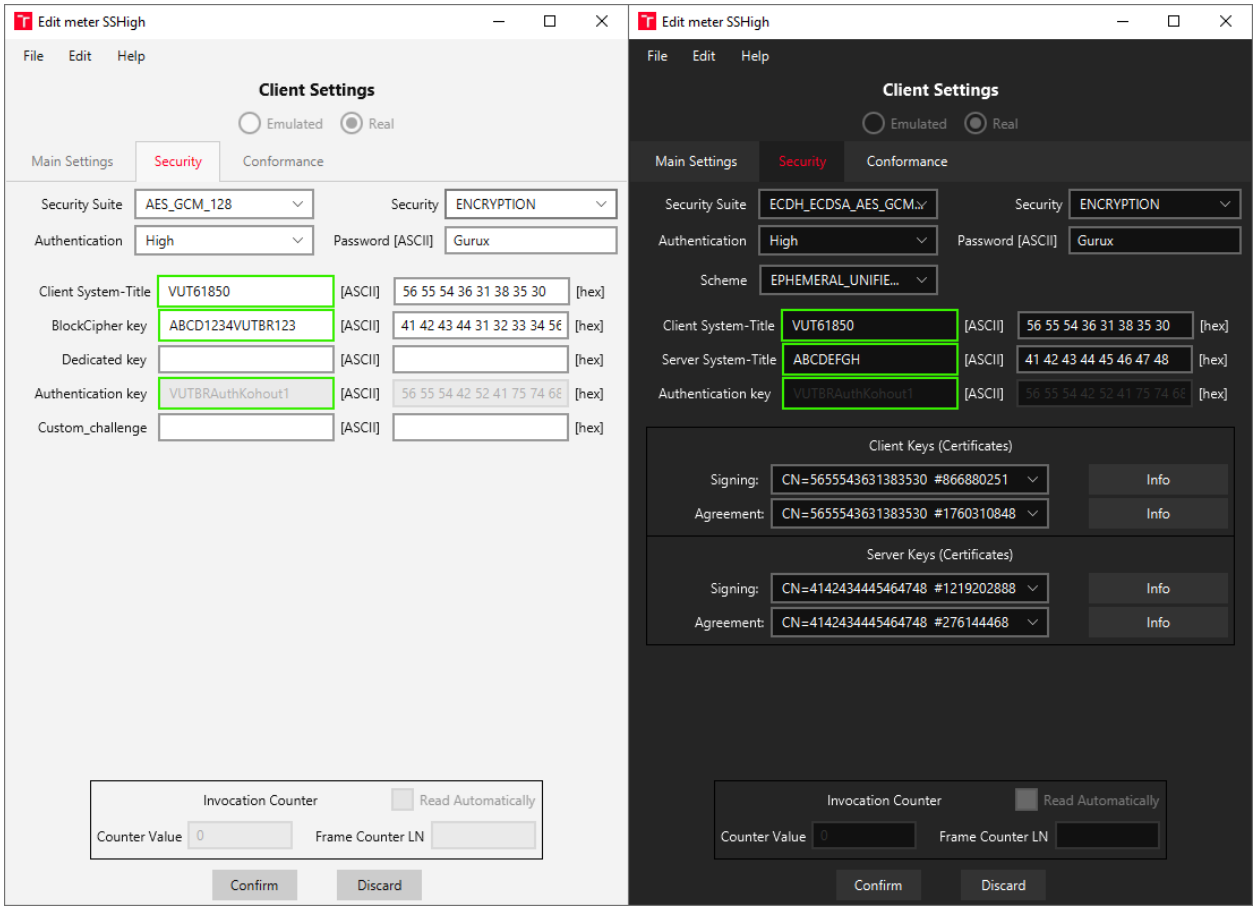


Obr. 5.4: Nastavení parametrů spojení

5.1.5 Pokročilé nastavení zabezpečení

Na obrázku 5.5 jsou znázorněna dvě okna. Okno na levé straně obsahuje nastavení zabezpečení pro `Security Suite 0` a okno na pravé straně obsahuje nastavení pro implementované zabezpečení dle DLMS `Security Suite 1 a 2`. Druhé okno má aktivní tmavé rozhraní, které je zde pro porovnání se světlým rozhraním.

Obě okna mají první 4 ovládací prvky společné. Typ autentizace, heslo a způsob zabezpečení jsou shodné s ovládáním v první záložce, navíc je zde nastavení security suite, podle kterého se přizpůsobuje zbytek rozhraní. Dalším společným prvek je na spodní straně okna, tzv. `invocation counter`, jehož nastavení je v rozhraní umístěno až pro budoucí použití, jelikož má vliv pouze na některé elektroměry. Zároveň je hodnota tohoto čítače důležitá při šifrovaní, avšak u testovaných zařízení jeho hodnota nemusela být měněna.



Obr. 5.5: Pokročilé nastavení zabezpečení

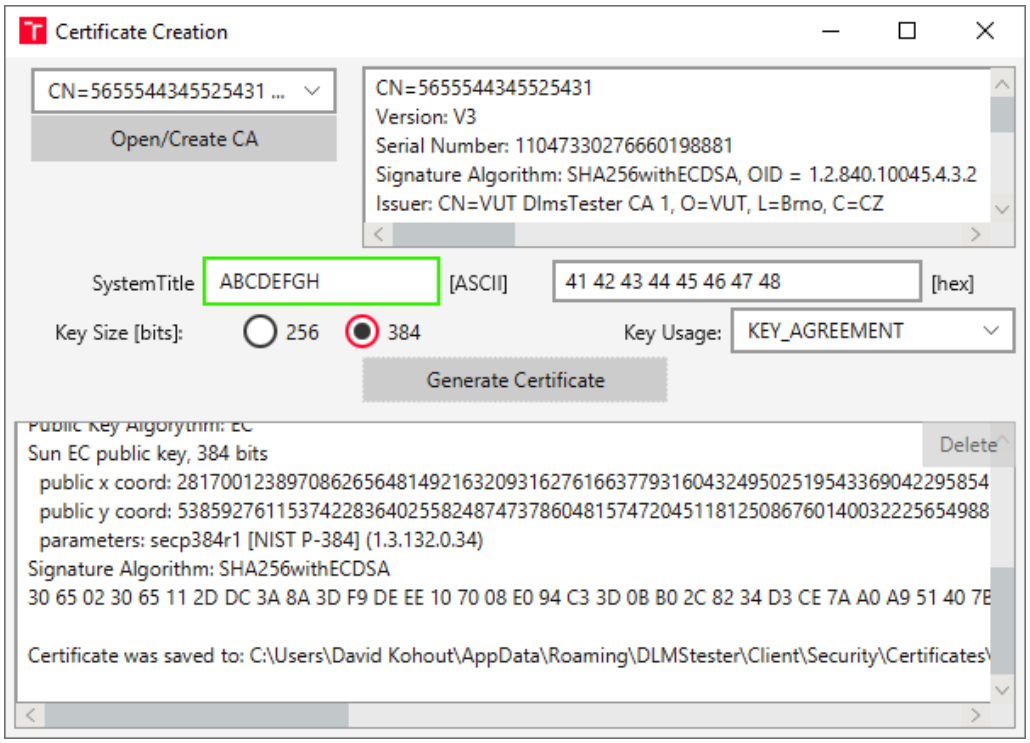
V okně pro `SS0` je nastavení označení klientské strany, dále několik klíčů (šifrovací, autentizační a dedikovaný) a nakonec možnost nastavení vlastní výzvy od klienta k serveru. Okno pro `SS1/2` obsahuje typ schématu pro určení, jakým způsobem dojde

k dohodnutí šifrovacího klíče, dále je zde označení klientské i serverové strany. Následuje autentizační klíč, který je shodný s klíčem u `SS0`. Nakonec je zde výběr vhodných certifikátů. Certifikáty mohou být podpisové a pro dohodnutí na klíči. Správné certifikáty se volí podle označení zařízení (`system-title`), podle účelu a případně i podle správného sériového čísla certifikátu.

5.1.6 Generování nových certifikátů

Pro účely testování je možné v aplikaci generovat vlastní certifikáty, které jsou podepsané vlastní certifikační autoritou. Jednoduché okno pro tvorbu certifikátů použitelných v DLMS je znázorněno na obrázku 5.6. Na horní straně se nachází výběr certifikační autority a základní údaje o zvolené autoritě. V případě potřeby je možné vytvořit novou autoritu nebo ji načíst z jiného umístění.

Certifikát se vydává pro určitý `System-Title`, a proto je možné jej specifikovat v tomto rozhraní. Dalším parametrem je velikost klíčů a tomu odpovídá také vhodná eliptická křivka. Klíče velikosti 256 bitů se využijí u `SS1` a velikost 384 bitů se použije u `SS2`. Posledním parametrem pro tvorbu certifikátu je použití klíčů. Tím může být účel podpisu nebo dohodnutí klíčů. Po navolení všech parametrů lze vygenerovat certifikát, který se uloží do aplikační složky.

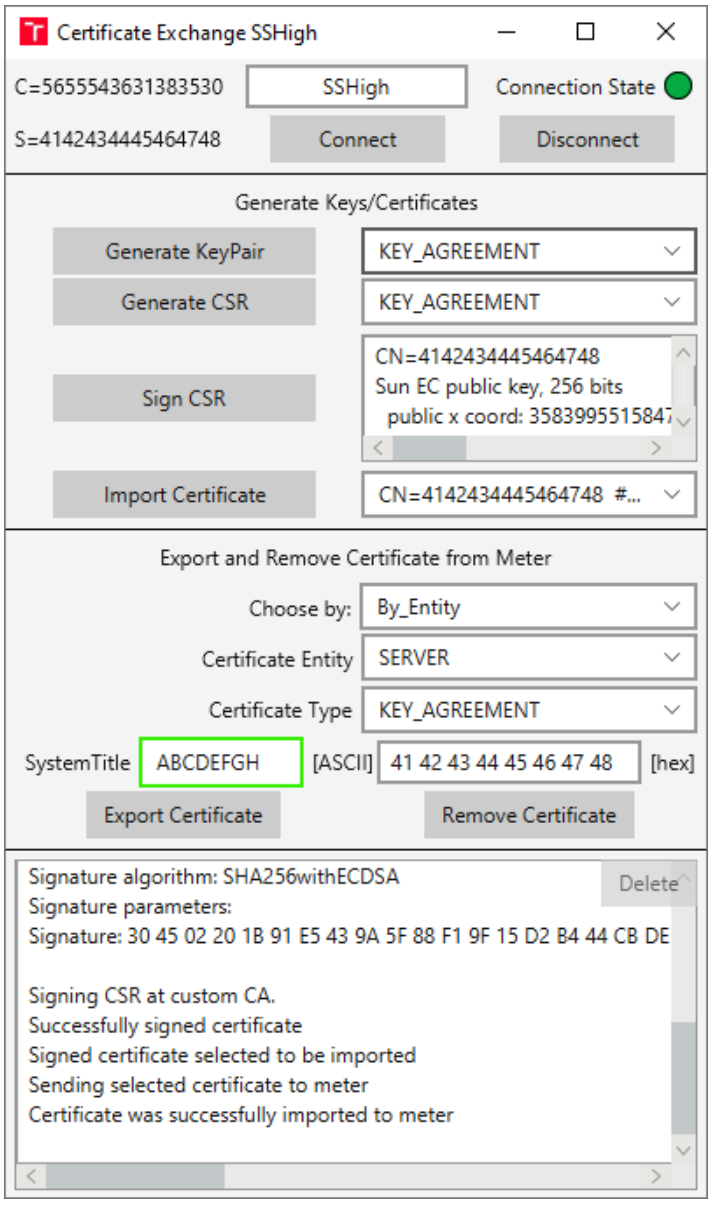


Obr. 5.6: Generování vlastních certifikátů

5.1.7 Výměna certifikátů s elektroměrem

Pokud elektroměr nebo klientská strana neobsahuje všechny potřebné certifikáty obou stran, je možné využít okno z obrázku 5.7 pro jejich výměnu. Na horní straně je vlastní název elektroměru a dále uložené `System-Title` v hexadecimálním formátu jak pro klientskou stranu „C=“, tak pro stranu elektroměru/serveru „S=“. Dále se zde nachází tlačítka pro navázání a ukončení spojení a indikátor připojení.

Pomocí tohoto rozhraní je možné provést komunikaci znázorněnou na obrázku 3.2.



Obr. 5.7: Výměna certifikátů s elektroměrem

U každého ovládacího tlačítka se nachází volba požadovaných parametrů pro danou funkci. `Generate KeyPair` vygeneruje na straně elektroměru nový pár klíčů pro nastavené použití. Následně je možné zaslat požadavek o vytvoření žádosti o certifikát

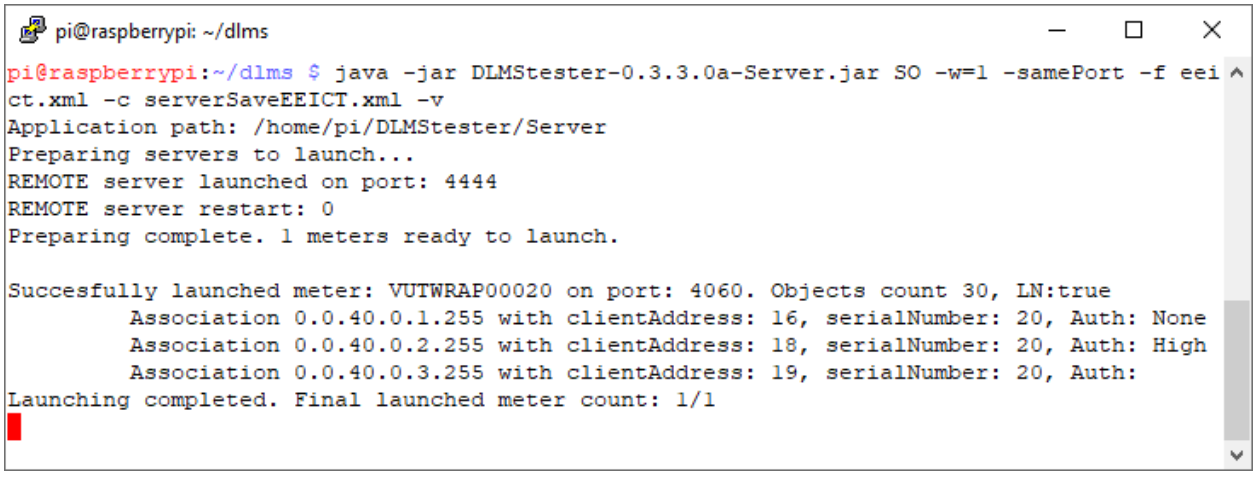
tlačítkem `Generate CSR`. Tímto krokem elektroměr vytvoří žádost s veřejným klíčem a zašle ji klientské straně, kde lze tuto žádost dále zpracovávat. Reálně by se žádost odeslala vydavateli certifikátů, kde by došlo k jeho vystavení. V případě aplikace se provede tvorba a podpis certifikátu pomocí vlastní certifikační autority. Jakmile je certifikát vydán, lze jej odeslat na stranu serveru pomocí `Import Certificate`. Pomocí této funkce lze zasílat i ostatní certifikáty do uložiště elektroměru.

Poslední možností při práci s certifikáty na dálku je jejich mazání a export na klientskou stranu. Obě tyto funkce požadují stejné parametry. Certifikát lze vybrat pouze pomocí jeho sériového čísla a vydavatele nebo složitěji pomocí více parametrů. Tento typ výběru se označuje jako `By_Entity` a vyžaduje typ certifikátu[2], dále typ entity[3] a posledním parametrem je `System-Title` certifikátu.

5.1.8 Implementace na Raspberry Pi

Díky využití jazyku Java není aplikace vázaná na konkrétní operační systém, a tak je možné aplikaci používat i například na RPi. Jedinou podmínkou je nainstalování Javy 11 na daném systému. Pro použití klientské aplikace je také vyžadována přítomnost grafického rozhraní. Z tohoto důvodu je pro systémy s čistě konzolovým rozhraním určená pouze aplikace serveru.

Použití emulovaného elektroměru pomocí vlastní aplikace je výhodné především pro testování. Aplikace díky tomu nemusí pracovat s tak omezeným hardware jako reálný elektroměr a také lze provádět úpravy softwaru, jako například přizpůsobení se novému přenosovému médiu či fungovaní nějakého elektroměru. Na obrázku 5.8 je znázorněna spuštěná aplikace pro server, která běží na RPi a lze se na jednotlivé emulované elektroměry připojit pomocí klientské aplikace.



Obr. 5.8: Spuštění emulovaného serveru

[2]Podpis, dohodnutí klíče, TLS nebo ostatní
[3]Server, klient, CA nebo ostatní

5.1.9 Použité knihovny a nástroje

S rozsáhlostí aplikace souvisí i počet použitých knihoven. Kromě DLMS části od Guruxu (viz kapitola 5.2) aplikace používá následující knihovny:

- JavaFX (15.0.1)
  - tvorba uživatelského rozhraní
- Jmetro (11.6.14)
  - moderní vzhled pro JavaFX rozhraní
- JAXB (2.3.1)
  - tvorba XML souborů pro ukládání konfigurací
- PicoCLI (4.6.1)
  - konfigurace příkazů a přepínačů v parametrech pro spuštění programu
  - využité jen pro spuštění serveru
- Commons-IO (2.8.0)
  - snadná práce se jmény souborů, získání koncovek apod.

Kromě knihoven byly k vývoji a testování použity tyto nástroje:

- Vývojové prostředí Eclipse IDE 2020-03 [18]
- Kompilátor Java JDK-11.0.5 [19]
- Nástroj Maven 4.0.0 [4]
- Program JavaFX SceneBuilder – 15.0.1 [20]
- Program Wireshark 3.4.4 [21]
- Doplněk pro Wireshark pro analýzu DLMS[5] [22]

5.2 Gurux DLMS knihovna pro Javu

Gurux je finská společnost, která se zabývá protokolem DLMS. Poskytují knihovny pro několik programovacích jazyků, mezi které patří například Python, C++, C# a Java. Právě knihovna pro Javu obsahuje nejvíce součástí z DLMS specifikace.

Gurux poskytuje všechny svoje produkty pod dvojí licencí [16]. Jeden druh licence je placený a poskytuje více funkcí a lepší podporu. Pro účely této práce je využita open source knihovna, na kterou se vztahuje licence GNU GPL v2 [17].

Celá knihovna lze rozdělit do 4 základních částí, které lze získat samostatně:

- gurux.dlms – nejdůležitější část, která implementuje funkčnost DLMS
- gurux.common – poskytuje třídy společné pro přenosová média (net, serial)
- gurux.net – třídy poskytující vhodnou spolupráci knihovny s TCP/IP
- gurux.serial – třídy pro komunikaci pomocí sériových linek

[4] Distribuováno s Eclipse a knihovny jsou dostupné přes Maven jako „dependencies“

[5] Vytvořil Ing. Petr Matoušek, Ph.D. VUT FIT pro svoji technickou zprávu [23]

6 Měření datových objemů

Zjištění vlivu zabezpečení na datový objem je důležité především z důvodů uvažovaných přenosových technologií, které mají omezenou datovou propustnost. Mezi tyto technologie patří například `NB-IoT`, `PLC`, `LTE Cat. M1` nebo `GPRS`. S objemem dat také souvisí cena za přenos těchto dat vzhledem k datovým tarifům.

Díky požadavkům na zabezpečení přenosu dat u technologií pro Smart Metering, specifikovaných v příloze č. 4 vyhlášky č. 365/2020 Sb. [1], se do budoucna musí počítat s rostoucím objemem dat způsobeným jejich šifrováním.

6.1 Nastavení měření

Pro měření objemů byla použita vyvinutá aplikace VUT DLMS Tester. Pro stranu elektroměru bylo využito emulovaného elektroměru za pomoci serveru, který je vyvíjen společně s klientskou aplikací.

Měření bylo plánované s využitím různých úrovní přístupu:

1. Nezabezpečená – kontrolní
2. Základní HLS autentizace
3. HLS s EC-DSA
4. Základní HLS + Security Suite 0
5. *Základní HLS + Security Suite 1
6. *Základní HLS + Security Suite 2

Pro testy s využitím jednotlivých sad, je možné provést dodatečný test, kde lze měnit účel zabezpečení na:

- Pouze autentizované zprávy
- Pouze šifrované zprávy
- Kombinace šifrovaných a autentizovaných zpráv

Při testování však bylo zjištěno, že není rozdíl v objemu dat, pokud je zpráva jen autentizovaná, nebo šifrovaná a autentizovaná. Toto zjištění také potvrzuje testovací příklad ze specifikace [5], který je uveden v příloze A. V příloze je patrné, že jakkoli zabezpečená data se vkládají jako obsah do zprávy jiného typu (s jiným příkazem). Nová zpráva pak následně obsahuje určitý tag, délku nesených dat, zabezpečovací hlavičku (kapitola 3.4.4) a až následně původní data (ať už šifrovaná či nikoliv) a v případě autentizovaných zpráv se za takto vytvořenou zprávu ještě vkládá autentizační tag.

*Tyto testy nemohly být otestovány z důvodů uvedených v kapitole 6.2.1.

Vyčítané objekty odpovídají požadovaným objektům, které jsou distributory plánované pro dálkový sběr v rámci AMM (Automatic Meter Management). Konkrétně se jedná o objekty znázorněné v následující tabulce 6.1. Z tabulky je patrné, že se celkově jedná o 21 objektů s 81 atributy.

Tab. 6.1: Testovací objekty

Typ objektu	**Počet atributů**	**Celkem objektů**	**Celkem atributů**
Register	3	18	54
Demand Register	9	3	27
Celkem		**21**	**81**

6.2 Porovnání objemů

V následujících grafech jsou vždy porovnány datové objemy různých částí komunikace. Získaná data pochází ze zachycené komunikace programem Wireshark, který umožňuje zjistit velikost celé TCP relace a režie DLMS protokolu. V grafech jsou využitá označení jako vysoká úroveň autentizace (**HLS**), dále označení Security Suite sady (**SS0**, **SS1** nebo **SS2**) a označení, zdali byla komunikace šifrovaná (**E**), jednotlivé zprávy byly autentizované (**A**), případně kombinace šifrování i autentizace (**EA**). V jednom případě je ještě konkretizovaná HLS autentizace s využitím digitálního podpisu EC-DSA. Pokud není uvedeno jinak, tak naměřený počet bajtů odpovídá velikosti celého spojení včetně TCP a nižších vrstev.

6.2.1 Testování SS1 a SS2

V době testování aplikace a určování datových objemů, bohužel nebyl k dispozici elektroměr, který by byl schopen komunikace právě s využitím vyššího zabezpečení pomocí alespoň Security Suite 1. Pro testování ostatních úrovní byl použit vlastní server, který vychází ze serveru od Guruxu 5.2, avšak v knihovně se nachází několik chyb, které neumožňují 100 % funkční komunikaci s využitím Security Suite 1 a 2.

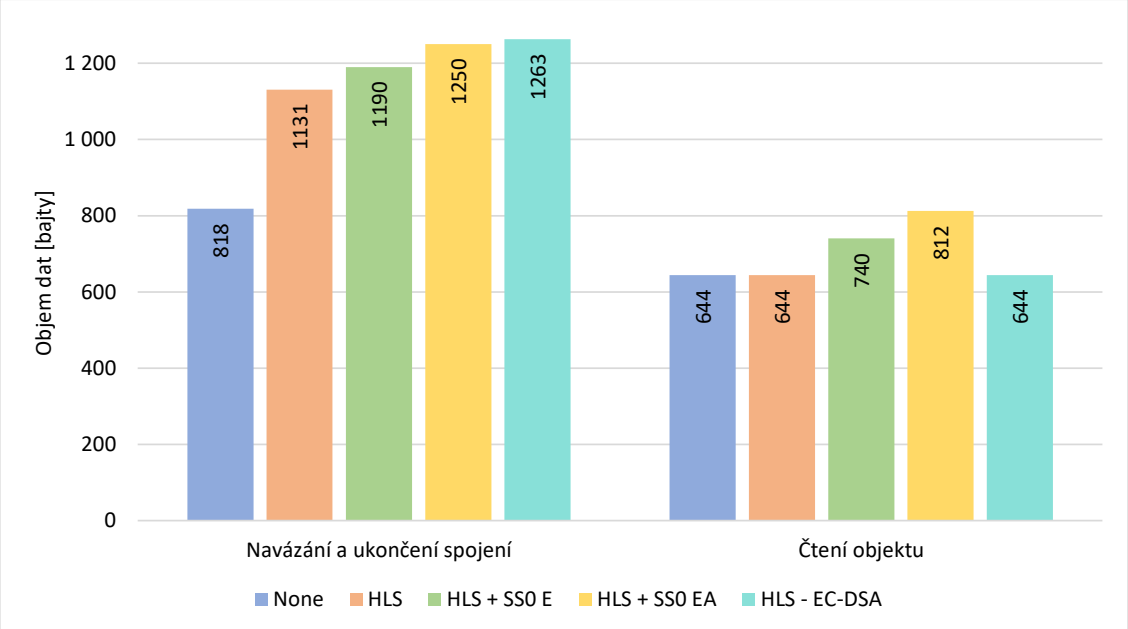
Hlavním problémem je ustanovení symetrických klíčů pro šifrování dat mezi klientem a serverem. Při navazování spojení se každá zpráva již zabezpečuje pomocí derivovaných klíčů ze známých údajů, jako jsou `System-Title` obou stran, `invocation couter` a také z klíčů certifikátů určených pro dohodnutí klíče. Toto derivování funguje bezchybně na prvních 3 odeslaných zprávách. Avšak pro poslední zprávu v průběhu AA dochází na každé straně k derivaci jiného klíče. Pokud by klíče byly shodné, tak by po poslední, čili čtvrté zprávě již došlo k dohodnutí symetrického klíče a zbytek komunikace by probíhal bezchybně.

6.2.2 Porovnání částí spojení

Na obrázku 6.1 je porovnán datový objem v bajtech, kde na levé straně jsou uvedeny velikosti pouze z navázání a ukončení spojení a sloupce na pravé straně uvádějí datovou velikost pouze ze čtení 1 objektu typu Register. Z grafu je patrné, že HLS autentizace navyšuje datovým objem především díky dodatečným 2 zprávám, které se při navázání spojení zasílají. Samotný typ HLS autentizace má pouze minimální vliv v rozmezí 5-12 %. Při porovnání s nezabezpečenou komunikací je tento datový nárůst v rozmezí 38-54 %.

Z druhé sady sloupců je patrné, že samotná autentizace spojení nemá vliv na čtení zpráv. Proto jediný nárůst způsobuje šifrování, případně šifrování v kombinaci s autentizováním zpráv. Samotné šifrování (SS0 E) způsobuje přibližně 15% nárůst a kombinace (SS0 EA) představuje 26% nárůst objemu u samotného čtení. U autentizovaných zpráv se na konec zprávy vkládá autentizační tag, který by měl pro každou zprávu být 12 bajtů dlouhý. Pokud tedy porovnáme 740 bajtů u šifrovaného a 812 bajtů u autentizovaného čtení, můžeme ověřit platnost zmíněného tvrzení. Musíme však vzít v potaz, kolik zpráv bylo při tomto čtení zasláno. Register objekt obsahuje 3 atributy, tím pádem se jedná o 3 žádosti a 3 odpovědi. Následně lze ověřit, že platí výpočet: $12 \cdot 6 + 740 = 812$.

U dosud nejvyšší autentizační metody za použití certifikátů a digitálního podpisu, představuje navázání spojení 54% nárůst dat oproti nezabezpečenému spojení, případně 12% nárůst při porovnání se základním HLS.

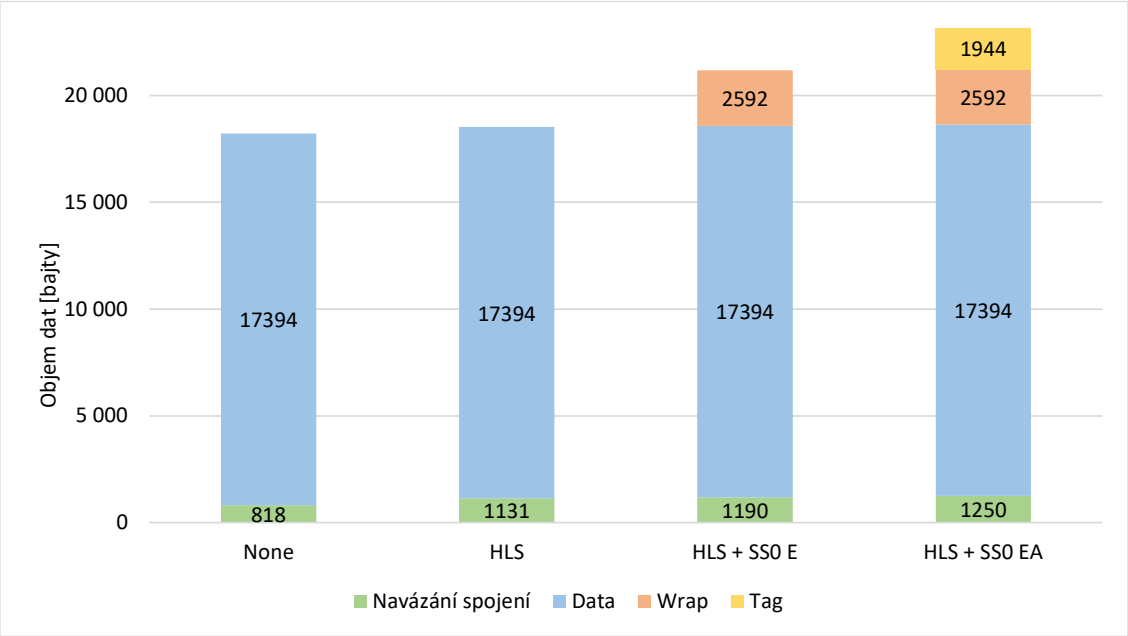


Obr. 6.1: Porovnání objemů pro část spojení

6.2.3 Porovnání složek spojení

Na obrázku 6.2 jsou uvedeny datové objemy z celého vyčítání všech 21 objektů. Celý objem je rozdělen na jednotlivé složky spojení (navázání a ukončení spojení, datová část, data vycházející ze šifrování (wrap) a autentizování zpráv (tag)). Data jsou při zabezpečení šifrováním, případně při autentizování zabalena do nové datové zprávy, kde tato nová zpráva obsahuje dodatečné informace, jako jsou nový příkaz, velikost vložených dat, zabezpečovací hlavička, čítač a také označení obou stran.

Data navíc u šifrování tvoří 12 % z celého objemu dat a v případě autentizovaných zpráv autentizační tagy tvoří 8 % z celého objemu. Opět zde lze provést výpočet pro ověření délky tohoto tagu. Jedná se zde o čtení všech 81 atributů, čili o 162 zpráv ze čtení. Pokud provedeme výpočet $12 \cdot 162 = 1944$ získáme stejnou hodnotu, která byla naměřena při testu.



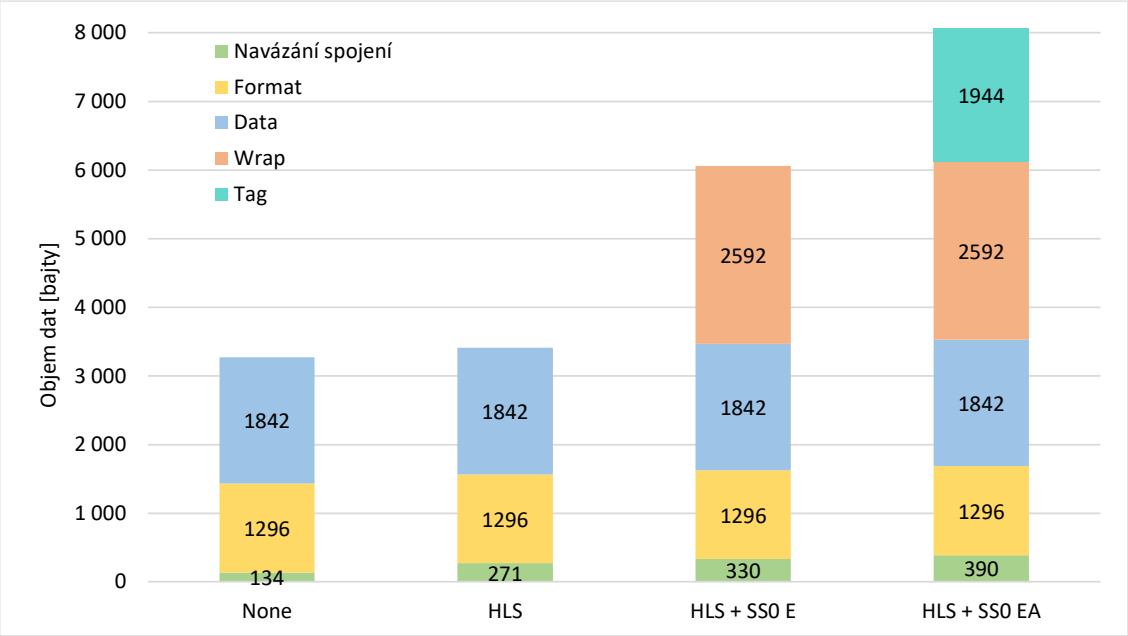
Obr. 6.2: Počet bajtů na části zpráv

6.2.4 Rozdělení spojení na DLMS části

Obrázek 6.3 obsahuje podrobnější rozdělení datových částí s ohledem pouze na DLMS data. Data jsou shodná s obrázkem 6.2, ale s absencí datových objemů způsobených samotnou přenosovou technologií. Tento graf zobrazuje objemy rozdělené do několika částí. Velikost u navázání spojení obsahuje také ukončení a formátování těchto zpráv. Další částí je formátování zpráv, kde se v tomto případě jedná o Wrapper, který v každé zprávě představuje 8 bajtů. Tento údaj lze ověřit vynásobením velikosti hlavičky WPDU s počtem zpráv: $8 \cdot 162 = 1296$. Zbytek částí je již shodný pro HDLC i Wrapper a to v případě použití odkazování pomocí logických jmen (LN).

Následuje část data, neboli APDU – samotná DLMS zpráva s příkazem a samotnými daty. Velikost datové části je shodná i při šifrovaní zpráv, protože využití AES-GCM má tu výhodu, že nevyžaduje padding [11] (výstupní velikost dat je shodná se vstupní velikostí). V případě zabezpečení zpráv se původní DLMS data vkládají do nového APDU, kde jsou tyto dodatečné informace v grafu označeny jako `Wrap`. Obsahují nový příkaz, velikost dat, `system-title` a bezpečnostní hlavičku. Poslední oddíl tvoří autentizování zpráv, které má vždy velikost 12 bajtů na jednu zprávu a v grafu je tato část označena jako `Tag`.

Z grafu lze odvodit, že SS0 přidává k šifrování dodatečných 16 bajtů informací ke každé zprávě, avšak průměrná velikost datové části je v případě tohoto testu jen 12 bajtů. To znamená, že šifrované zprávy jsou přibližně o 140 % větší.



Obr. 6.3: Rozdělení objemu na části DLMS

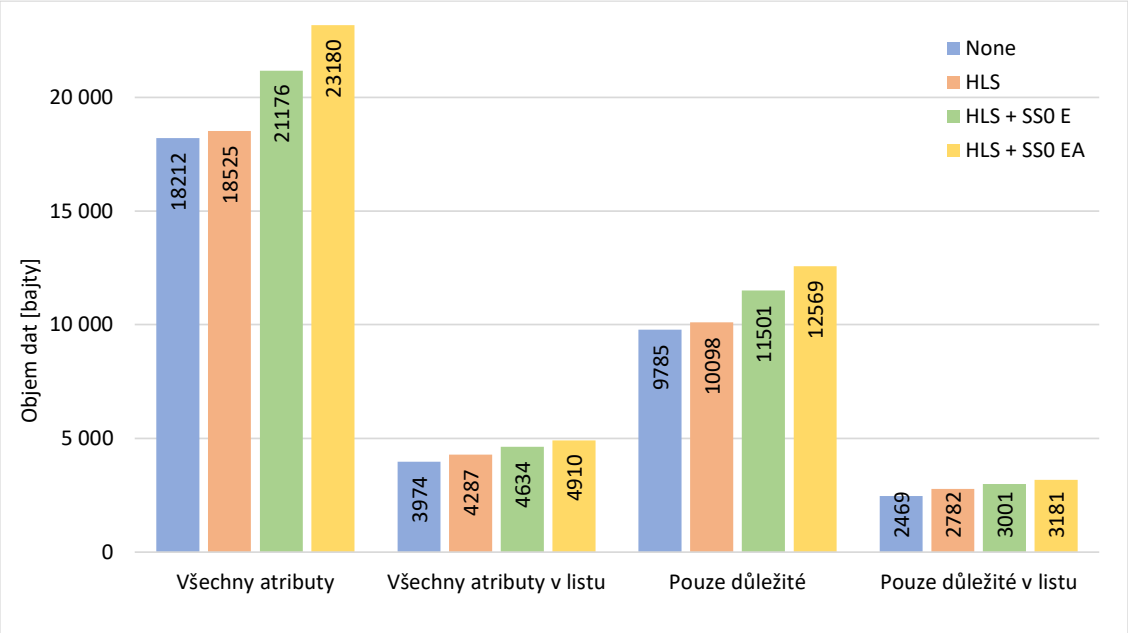
6.2.5 Porovnání způsobů čtení

Poslední obrázek 6.4 uvádí další způsoby, pomocí kterých lze vyčítat data z elektroměru. První sloupec znázorňuje čtení všech atributů, kde každý atribut znamená jednu žádost a jednu odpověď. Další sloupec uvádí opět čtení všech dat, ale jejich čtení se provádí za pomoci listu. V jednom listu se nachází 10 atributů a celé čtení 81 atributů je tak zmenšeno do 9 žádostí a odpovědí.

Poslední dva sloupce uvádí opět čtení zvlášť a za pomoci listu. Zde však nedochází ke čtení všech atributů, ale vyčítány jsou pouze důležité části. Například není potřebné číst označení objektu, jelikož tímto označením se na daný objekt dotazujeme a jeho hodnota je obsažena v každém objektu na 1. atributu. Další data,

která se nemusí číst pokaždé, jsou například jednotky a měřítko dat, jelikož jsou tyto údaje často statické. Například u každého `Register` objektu stačí číst pouze jeden atribut. Celkově se tak v případě testu jedná pouze o 42 atributů.

Všechny tyto metody umožňují snížení celkového objemu dat při čtení. Z předchozích grafů je patrné, že největší část datového objemu způsobuje samotný přenos dat a použité přenosové technologie. Pro snížení datového objemu je tak nutné minimalizovat počet zasílaných zpráv, aby byla datová režie co nejmenší. Konkrétně u nezabezpečeného vyčítání způsobuje čtení v listu téměř pětinásobnou úsporu datového objemu.



Obr. 6.4: Porovnání objemů při rozdílném typu čtení

6.2.6 Celkové vyhodnocení

Při porovnání datového objemu u čtení bez zabezpečení byl celkový přenesený objem 18 218 bajtů, ale objem DLMS části je pouze 3 272 bajtů. Z toho je patrné, že samotný přenos dat potřebuje téměř 5x více objemu než samotná DLMS část.

Dodatečná data, která se přidávají u šifrování a autentizování zpráv při použití `Security Suite 0` představují dodatečných 16 bajtů a při použití autentizace se přidává dalších 12 bajtů pro autentizační tag.

Pro maximální úsporu dat je však nejvhodnější zasílat malé množství velkých zpráv, tedy použít čtení v listu. To dosahuje téměř 80% úspory dat oproti čtení zpráv zvlášť.

Závěr

Tato diplomová práce byla zaměřena především na problematiku zabezpečení ve standardu DLMS a určení vlivu zabezpečení na datové objemy.

Po úvodním seznámením s možnostmi využití DLMS, jsou v práci sepsány některé základy se kterými se lze s tímto protokolem setkat. Především se jedná o vysvětlení problematiky adresování v kapitole 2.1.

Další kapitola 3 se již věnovala zabezpečení v DLMS, které je rozděleno do dvou základních částí a to autentizace (kapitola 3.2) a šifrování (kapitola 3.4). Pro část šifrování jsou ještě představeny zabezpečovací sady a pro vyšší úroveň zabezpečení jsou přestaveny použité algoritmy, certifikáty (kapitola 3.5.2) a způsoby pro dohodnutí symetrického klíče v kapitole 3.6. U certifikátů je také představena vzdálená práce s certifikáty na straně elektroměru pomocí určitých příkazů.

Následně pro účely testování bylo potřeba zjistit, jaké objemy dat se v DLMS zasílají. Z tohoto důvodu je v práci prakticky ukázáno, jak se vytváří a liší dva typy formátování zpráv (HDLC x Wrapper) v kapitole 4.1.

V praktické části je představena vlastní aplikace pro testování VUT DLMS Tester (kapitola 5.1). Jednotlivé prvky uživatelského rozhraní jsou doplněné krátkým popisem funkčnosti daného okna. Kromě základní funkčnosti byla aplikace rozšířena o možnosti testování vyšších úrovní zabezpečení pomocí Security Suite 1 a 2, včetně tvorby vlastních certifikátů a způsobu jejich výměny mezi klientem a serverem. Následně jsou zmíněny použité nástroje včetně DLMS knihovny Gurux (kapitola 5.2).

Funkčnost VUT DLMS Testeru byla otestována komunikací s reálnými elektroměry několika výrobců jako jsou ZPA, Landis+Gyr a Itron. Následně pro další testování byla vytvořena vlastní aplikace pro emulování elektroměrů a tato aplikace byla funkčně otestovaná na Raspberry Pi.

Poslední kapitola 6 se zaměřuje na testování datových objemů s využitím distributory předpokládaných dat pro dálkové odečty. Na těchto datech jsou provedeny testy s využitím různých úrovní přístupu a výsledná data jsou zanesena do přehledných grafů. Testování a výsledky měření byly konzultovány s výrobcem elektroměrů ZPA a s distribuční společností EG.D.

Z provedeného měření lze vyvodit, že zabezpečením samotných DLMS dat nastává více než dvojnásobný datový nárůst oproti nezabezpečeným datům (u čtení atributů zvlášť). Pokud se jedná o celkový objem zaslaných zpráv, včetně dat nižších vrstev, šifrované zprávy jsou přibližně o 20 % větší než u nezabezpečeného spojení.

Výsledky diplomové práce byly prezentovány na konferenci „Student EEICT 2021“ [24]. Do budoucna je možné aplikaci rozšířit například o kompresi dat a provést další měření s elektroměry, které podporují Security Suite 1 a 2.

Literatura

[1] *Vyhláška o měření elektřiny: Vyhláška č. 359/2020 Sb.* [cit. 2021-03-01] Dostupné také z: https://www.zakonyprolidi.cz/cs/2020-359

[2] MOLEK, Tomáš. Smart region Vrchlabí. *oEnergetice* [online]. [cit. 2021-03-01]. Dostupné z: http://oenergetice.cz/elektrina/smart-region-vrchlabi-prvni-ceska-chytra-sit/

[3] Projekt SMARAGD: Zavádění chytrého měření. *EG.D* [online]. [cit. 2021-03-01]. Dostupné z: https://www.egd.cz/smaragd

[4] *DLMS UA* [online]. [cit. 2020-12-02]. Dostupné z: https://www.dlms.com/

[5] *Green Book: DLMS/COSEM Architecture and Protocols*. Ed10–V1.0. Switzerland: DLMS User Association, 2020.

[6] *Blue Book: COSEM Interface Classes and OBIS Object identification System*. Ed14–V1.0. Switzerland: DLMS User Association, 2020.

[7] Norma *IEC 62056* [online]. IEC Webstore [cit. 2020-12-02]. Dostupné z: https://webstore.iec.ch/searchform&q=IEC%2062056

[8] KOHOUT, David. *Zátěžový generátor zpráv DLMS/COSEM*. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií, 2019.

[9] DLMS Client and Server addressing. *Gurux* [online]. [cit. 2020-12-02]. Dostupné z: https://www.gurux.fi/dlmsAddress

[10] *ITU-T V.44: Data compression procedures*. ITU-T, 2000. Dostupné také z: https://www.itu.int/rec/T-REC-V.44-200011-I

[11] RFC 5084: AES-CCM and AES-GCM in CMS. *IETF* [online]. [cit. 2021-5-6]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc5084#section-3.2

[12] FIPS 186-4: Digital Signature Standard (DSS). *NIST* [online]. [cit. 2021-5-6]. Dostupné z: https://csrc.nist.gov/publications/detail/fips/186/4/final

[13] NIST SP 800-56A Rev. 2: 2013.: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography. *NIST* [online]. [cit. 2021-5-6]. Dostupné z: https://csrc.nist.gov/publications/detail/sp/800-56a/rev-2/archive/2013-06-05

[14] SP 800-56A Rev. 3: Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography. *NIST* [online]. [cit. 2021-5-6]. Dostupné z: https://csrc.nist.gov/publications/detail/sp/800-56a/rev-3/final

[15] Information Technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures. *ISO/IEC 13239:2002*. ISO/IEC 1, 2001.

[16] *Dual licensing of Gurux products* [online]. Gurux [cit. 2020-12-01]. Dostupné z: https://www.gurux.fi/duallicense

[17] *Obecná veřejná licence GNU v.2* [online]. GNU GPL Czech [cit. 2020-12-01]. Dostupné z: http://www.gnugpl.cz/v2/

[18] Vývojové prostředí *Eclipse* [online]. The Eclipse Foundation [cit. 2021-02-22]. Dostupné z: https://www.eclipse.org/downloads/packages/release/2020-03/r

[19] *Java SE Development Kit 11* [online]. Oracle [cit. 2021-02-22]. Dostupné z: https://www.oracle.com/java/technologies/javase-jdk11-downloads.html

[20] Nástroj pro tvorbu grafického rozhraní *Scene Builder* [online]. Gluon [cit. 2021-02-22]. Dostupné z: https://gluonhq.com/products/scene-builder/

[21] *Wireshark* [online]. [cit. 2021-02-22]. Dostupné z: https://www.wireshark.org/

[22] Doplněk pro Wireshark *DLMS analysis* [online]. GitHub [cit. 2021-02-22]. Dostupné z: https://github.com/matousp/dlms-analysis

[23] MATOUŠEK, P. Technical report: *Analysis of DLMS Protocol* [online]. VUT FIT-TR-2017-13, Brno, CZ, 2017 [cit. 2021-02-22]. Dostupné z: https://www.fit.vut.cz/research/publication/11616.

[24] KOHOUT, D. Impact of Cyber Security on Data Volume in Smart Metering. *In Proceedings of the 27th Conference STUDENT EEICT 2021.* Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií: 2021. ISBN: 978-80-214-5942-7.

Seznam symbolů a zkratek

AA	Application Associations
AARE	Application Associations Response
AARL	Application Associations Release
AARQ	Application Associations Request
AES	Advanced Encryption Standard
APDU	Application Protocol Data Unit
CN	Common Name
COSEM	Companion Specification for Energy Metering
CSR	Certificate Signing Request
CtoS	Client to Server
DLMS	Device Language Message Specification
DLMS UA	DLMS User Association
ECDH	Eliptic Curve Diffie-Hellman
ECDSA	Eliptic Curve Digital Signature Algorithm
GCM	Galois/Counter Mode
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HDO	Hromadné dálkové ovládání
HLS	High Level Security
IoT	Internet of Things
LLC	Logical Link Control
LLS	Low Level Security
LN	Logical Name
LSB	Least Significat Bit

LTE	Long-Term Evolution
MAC	Media Access Control
NB-IoT	Narrowband - Internet of Things
OBIS	OBject Identification System
PDU	Protocol Data Unit
PLC	PowerLine Communication
RPi	RaspberryPi
SAP	Service Access Point
SN	Short Name
SS1/2	Security Suite 1/2
StoC	Server to Client
TCP	Transmission Control Protocol
WPDU	Wrapper Protocol Data Unit

A Příklad s autentizací a šifrováním

	X	Contents			*LEN (X) bytes*	*Len (X) bits*
Security material						
Security suite		GCM-AES-128				
System Title	*Sys-T*	**4D4D4D**0000BC614E(here, the five last octets contain the manufacturing number in hexa)			8	64
Invocation counter	*IC*	01234567			4	32
Initialization Vector	*IV*	*Sys-T* ‖ *IC* 4D4D4D0000BC614E01234567			12	96
Block cipher key (global)	*EK*	000102030405060708090A0B0C0D0E0F			16	128
Authentication Key	*AK*	D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF			16	128
Security applied		**Authentication**	**Encryption**	**Authenticated encryption**		
Security control byte (with unicast key)	*SC*	*SC-A* 10	*SC-E* 20	*SC-AE* 30	1	8
Security header	*SH*	*SH = SC-A* ‖ *IC* 1001234567	*SH = SC-E* ‖ *IC* 2001234567	*SH = SC-AE* ‖ *IC* 3001234567	5	40
Inputs		**Authentication**	**Encryption**	**Authenticated encryption**		
xDLMS APDU to be protected	*APDU*	C0010000080000010000FF0200 (Get-request, attribute 2 of the Clock object)			13	104
Plaintext	*P*	Null	C001000008000001 0000FF0200	C001000008000001 0000FF0200	13	104
Associated data	*A*	*SC* ‖ *AK* ‖ *APDU*	–	*SC* ‖ *AK*		
Associated Data – Authentication	*A-A*	10D0D1D2D3D4D5D6 D7D8D9DADBDCDDDE DFC0010000080000 010000FF0200	–	–	30	240
Associated Data – Encryption	*A-E*	–	–	–	0	0
Associated Data – Authenticated encryption	*A-AE*	–	–	30D0D1D2D3D4D5D6 D7D8D9DADBDCDDDE DF	17	136
Outputs		**Authentication**	**Encryption**	**Authenticated encryption**		
Ciphertext	*C*	NULL	411312FF935A4756 6827C467BC	411312FF935A4756 6827C467BC	13	104
Authentication tag	*T*	06725D910F9221D2 63877516	–	7D825C3BE4A77C3F CC056B6B	12	96
The complete Ciphered APDU		*TAG* ‖ *LEN* ‖ *SH* ‖ *APDU* ‖ *T*	*TAG* ‖ *LEN* ‖ *SH* ‖ *C*	*TAG* ‖ *LEN* ‖ *SH* ‖ *C* ‖ *T*	–	–
Authenticated APDU		C81E1001234567C0 0100000800000100 00FF020006725D91 0F9221D263877516	–	–	32	256
Encrypted APDU		–	C8122001234567411 312FF935A475668 27C467BC	–	20	160
Authenticated and encrypted APDU		–	–	C81E300123456741 1312FF935A475668 27C467BC7D825C3B E4A77C3FCC056B6B	32	256

Převzato z Green Book [5], pro ukázku rozdílu použití autentizovaných a šifrovaných zpráv, včetně jejich kombinace.

B Obsah elektronických příloh

Elektronické přílohy obsahují logy a záchyty komunikace z testování. Vytvořená aplikace nemohla být do příloh zahrnuta z důvodů omezené kapacity příloh a proto v případě zájmu je možné kontaktovat autora práce pro zaslání aplikace.

- / ..... kořenový adresář elektronických příloh
  - Aplikace
    - Aplikace.txt
    - DPTestovani.xml ..... Konfigurace aplikace z testování
  - Logy
    - 1. None .....Složka obsahuje všechny typy čtení
    - 2. HLS .....Složka obsahuje všechny typy čtení
    - 3. HLS + SSO E .....Složka obsahuje všechny typy čtení
    - 4. HLS + SSO EA .....Složka obsahuje všechny typy čtení
    - 5. HLS - ECDSA .....Složka obsahuje všechny typy čtení
    - Navázání spojení.txt
  - Pouze DLMS zprávy
    - Celé HLS čtení (spojeno).txt
    - HLS - rozbor čtení všeho.txt
    - HLS + SSO E - rozbor čtení všeho.txt
    - Pouze data z navázání spojení.txt
    - Pouze otevřená data DLMS celého čtení (spojeno).txt
    - Pouze šifrovaná data DLMS celého čtení (spojeno).txt
    - Pouze zabalení šifrovaných dat DLMS celého čtení (spojeno).txt
  - Záchyty
    - Čtení jednoho objektu
      - HLS + SSO E.pcapng
      - HLS.pcapng
      - None.pcapng
    - Rozdělené
      - Čtení None, HLS, E postupně All, Alllist, H, Hlist.pcapng
      - ECDSA - Navázaní, čtení all.pcapng
      - HLS + SSOEA, All, AllList, H, Hlist.pcapng
      - Porovnání EA, E, A.pcapng
      - TestNavazani - None,HLS,E,EA,ECDSA.pcapng
    - 1. None všechny testy.pcapng
    - 2. HLS všechny testy.pcapng
    - 3. HLS SSO E všechny testy.pcapng
    - 4. HLS SSO EA všechny testy.pcapng
    - 5. HLS ECDSA všechny testy.pcapng
    - Pořadí testů v souborech.txt
- Implementace zabezpečení do DLMS protokolu.pdf .... Elektronická verze práce